Kyoto University

Doctoral Thesis

GIS-based Intelligent Assistant Agent for Supporting Decisions of Incident Commander in Disaster Response

Author:
Reza Nourjou

Supervisor:
Dr. Michinori Hatayama

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Disaster Prevention Research Institute
Department of Social Informatics, Graduate School of Informatics

February 2014

Declaration of Authorship

I, Reza NOURJOU, declare that this thesis titled, 'GIS-based Intelligent Assistant Agent for Supporting Decisions of Incident Commander in Disaster Response' and the work presented in it are my own. Signed:

Date:

Abstract

by Reza NOURJOU

Problem: In disaster emergency response management, the efficient coordination of a team of field units is an essential issue. It is crucial for the team in applying an intelligent software agent that assists an incident commander in action planning, task scheduling, and decision making in crisis response. Our motivation is to address this requirement.
Objective: This thesis intends to develop a GIS-based intelligent assistant agent that supports an incident commander' s decisions in coordination of disaster crisis response operations.
Method: The methodology is to define a big scope and decompose the main research question into several some research questions. Seven chapters are provided that each one addresses a challenging problem regrading to the main problem. Each chapter includes a significant contribution for a research question addressed by this chapter.
Results: This thesis achieved seven contributions. They include as follows: 1) design of a GICoordinator that integrates geoinformatics with artifice intelligent techniques in order to provide sufficient tools for support of an IC, 2) a data model that models and formulates the strategic planning and scheduling problem, 3) intelligent algorithms that aim to (I) automatically calculate a set of feasible alternatives for strategic decision making and (II) autonomously identify a subset of assigned agents that should be released from their threads in order to refine a strategic decision, 4) automated algorithm that aims to assign location-based temporal macro tasks to field unites that the result includes (I) a macro task/action schedule, (II) an overall execution time of tasks execution and, (III) a right time for revising a strategic decision, 5) A* search algorithm that aims to (I) optimize the the strategic decision-making problem by selecting the best choice from the a set of alternatives and (II) estimate a minimum overall time which the team needs to achieve the objective, 6) A simulator that provides a flexible and efficient framework for C# developers who want to develop, simulate, and evaluate community of a number of instances of the GICoordinator, and 7) simulate spatially distributed intelligent assistants that each one runs on a tablet computer and assists the field units in distributed task distribution/allocation among the team members.

Acknowledgements

I would like to appreciate Dear all who were beside me during these years, and without them I could not be here. Although their kind names is being mentioned here, each name reminds me of many good memories.

Specially, I would like to thank Dr. Hatayama, the supervisor of my PhD, and Prof. Tatano, the director of Social System for Disaster Risk Governance laboratory of DPRI for kindness, support, care, and concern they did for my progress and success. I like to appreciate Mrs. Atsuyo Yamasita for every thing she did for me.

I would like to thank Dr. Babak Mansouri, Prof. Mohsen Ghafory-Ashtiany, Prof. Noria Okada, Prof. Toru Ishida, Prof. Stephen F. Smith, Dr. Pedro Szekely, Maja Ostric, GCOE-HSE of Kyoto University, Seitarou Yukawa, good students of the Lab, Siddharth Mehrotra, Mr. Shigenori Shibata, Mrs. Shahrbanoo Amani, Mr. Rasoul Dadras, Kourosh Meshgi, Hadi Abroshan, Mohammad-hossein Tahersima, Prof. Avraham Melkman, Matteus Tanha, Ebrahim Meshkati Shahmirzadi, Tetsuya Tamaki, Daryoosh Haziq, Mrs. Florence lahournat, Dr. Kazuyoshi Nakano, Eisuke Imai, Megumi Anto, Yusuke Yoshimura, Sara dadras, Hossein Abrahimi, Hossein Aghamohammadi, Hamid-Reza Jafarian, Eehsan Borhani, Mr. Mineo Tokuda, Mrs. Midori Imamura, Mr. Abbas Mokabbery, Dr. Xiao-Feng Xie, Behnam Miripour Fard, Bahram Safaee, Toofan Nabizada, lassaad chourou, and Amir Nourjou.

To Soude

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Abstract

Problem: Efficient coordination of a team of field units is an essential issue in disaster emergency response management. It is crucial for the team in applying intelligent systems that assists/supports the IC (incident commander) and collaborates with human in this issue. Development of this IC support system requires a multidisciplinary approach of several sub-contributions.

Objective: The purpose of this chapter is to design requirements of spatial intelligent coordinators. It aims to state a number research questions that address significant challenges in the main research question.

Method: This chapter analyzed and discussed the responsibilities of an IC in centralized coordination of field units.

Results: Seven key research questions were proposed. Each one address a challenge that requires a contribution that is the considerable importance in developing an ideal solution.

Conclusion: A framework was defined as a road map that states what necessary contributions we should achieve in order to develop a GIS-based intelligent assistant agent that can support the IC in efficient coordination.

1.2 Background

Review of the past disasters shows that efficient coordination is an important and essential issue in emergency/crisis response operations. A responder team that comprises an

IC (incident commander) and several field/operational units is faced with the problem of carrying out spatially dispersed tasks under evolving execution circumstances in a manner that achieves a joint objective in a minimum time. Coordination is the act of managing interdependencies among activities performed to achieve a goal. The IC plays a crucial role in controlling and coordinating actions of field units because inefficient coordination results in idle agents, conflict between actions, or redundant activities.

Effective coordination is hard and challenging to be achieved because of the complexity of this domain. Disaster emergency response is characterized an a dynamic, spatial, uncertain environment that includes uncertainty and ambiguity information about tasks, uncertainty in outcomes of actions, multiple actors, time pressure, limited resources, task flow, distributed information, etc.

An IC support system is a complex system. Development of this system requires a multidisciplinary approach of planning, scheduling, simulation, system development, resource allocation, decision making techniques, etc. The focus point is on intelligent systems that assist human, especially the IC, and collaborate with him in problem solving.

In the recent decade, multi-agent systems have been used for crisis/emergency management systems.There is much literature in coordination of disaster emergency operations, and each one addresses a specific problem with specific requirements and assumption. According to this problem domain, We address a new problem with new ideas. Some research questions as research challenges are considerable to be addressed.

An ideal solution for the coordination problem is required a number of contributions/issues. It is important to identify these significant difficulties in the main research question. The purpose of this chapter is to design requirements of spatial intelligent coordinators by decomposing this research questions into several key research questions.

1.3 Motivation

Imagine that an earthquake disaster has occurred in an urban area which is displayed in Fig. 1.1. Urban search and rescue (USAR) aims to rescue the greatest number of people who are trapped under the debris of damaged buildings, which are spatially distributed in the area, in the shortest amount of time. To save a victim in a certain spatial location, a sequence of dependent tasks should be accomplished, and each type of task requires a set of certain capabilities, a considerable amount of time, or an amount of resources.

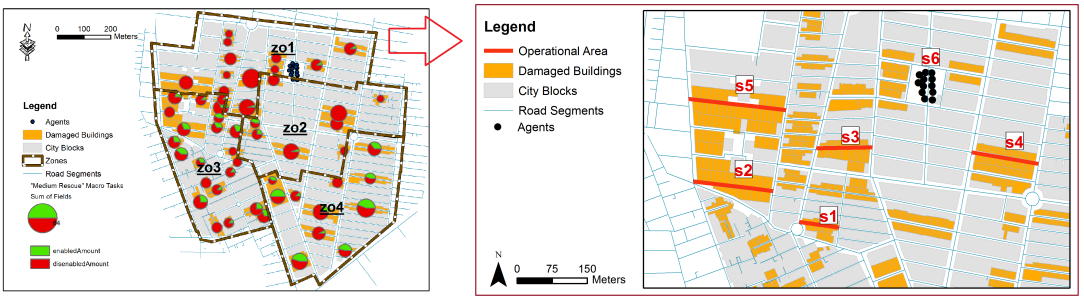


FIGURE 1.1: A GIS map of the initial state of the world.

A USAR team of four field units, which are called agents, has been assigned to five operational areas which are presented by five road segments displayed in Fig. 1.1. An agent can have a number of different actions that each action provides a set of capabilities which is required to do a task or several tasks simultaneously. It was assumed these agents are free (or idle) and are located in the incident command post.

Shortest distances among six road segments are calculated using GIS. To do spatially distributed tasks, agents need to move from one location to another through the road network with the moving speed equal to 20. These data are used by the IC in decision making. These data are provided and are updated by a relevant team or organization whose activities are to clear road blockages.

A set of LoTeM tasks (Location-based Temporal Macro tasks) are associated to each road segment. Simply, a LoTeM task is an aggregative task of all tasks with the same task type which are spatially located within a geographic area such as road segment, city block, etc [10]. Imagine that twelve LoTeM tasks are located to five segments at the time 0. Information of these tasks forms the big picture that the IC observes from the task environment.

Fig. 1.2 presents a flow work which is used by incident commanders for coordination of their teams. It states the action planning and scheduling techniques that include objective selection and decomposing it into sub-goals, grouping available units into coalitions and assigning them to the sub-goals, allocation of the units to tasks, and adaption of the made decisions.

The domain of disaster emergency/crisis response is a complex environment. Therefore, in order to maximize the global payoff, it is important to support the IC in this work flow. The main research question is that how to develop a spatial intelligent coordinator that supports/assists an IC in this work flow?

1.4 Literature Review

First, we appreciate all previous works that enabled us to define a new problem that includes some new ideas.

The incident commander system is a disaster management tool based on a series of rational bureaucratic principles for disaster responses [4]. Basic system objectives and plans are established at or near the top of the hierarchy and used as bases for decisions and behaviors at lower levels. Unfortunately, FEMA provides a set of useful manuals and guidelines about practices but it does not make explicit the design requirements and algorithms for making incident action plans.

DEFACTO incorporates artificial intelligence, 3D visualization and human-interaction reasoning into a unique high fidelity system for training incident commanders. Key aspect focuses on adjustable autonomy that refers to an agent‘s ability to dynamically change its own autonomy, possibly to transfer control over a decision to a human or another agent [16]. Adjustable autonomy is different with strategic planning problems which incident commanders try to solve.

DrillSim is a multi-agent simulation and modeling system for testing IT solutions; it aims to play out a disaster response activity where agents might be either computer agents or real people [1]. In ALADDIN, decentralized systems aim to gather information from variety of heterogeneous sources in order to take informed action; it considers different aspects such as data fusion, decision making, machine learning, and system architecture [5]. A spatio-temporal task allocation algorithm to human groups is proposed for rescue operations management [17]. The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams (fire-fighting, police, ambulance) acting in large urban disasters to to minimize damages caused by earthquake such as civilians buried, buildings on fire and blocked roads [7]. FireGrid is a task-centric collaborative community to pursue research for developing realtime response systems using the Grid; it addresses response process in the built environment, where sensor grids in large-scale buildings are linked to super-real time grid-based simulations [3]. The problem of distributed coordination among field units is addressed by the coordinators project [2]. WIPER is intends to provide emergency planners and responders with an integrated system that will help to detect possible emergencies, as well as to suggest and evaluate possible courses of action to deal with the emergency [15]. However, these works have significant contributions for the domain of emergency response management, unfortunately they do not address the whole problem which is addressed by this chapter.

1.5 Problem Statement

To get a better insight of the problem, Fig. 1.2 presents the work flow of coordination within a team during disaster emergency response operations. Our focus is on responsibilities of the IC of a team. The following subsections describe input data that make up the problem space.

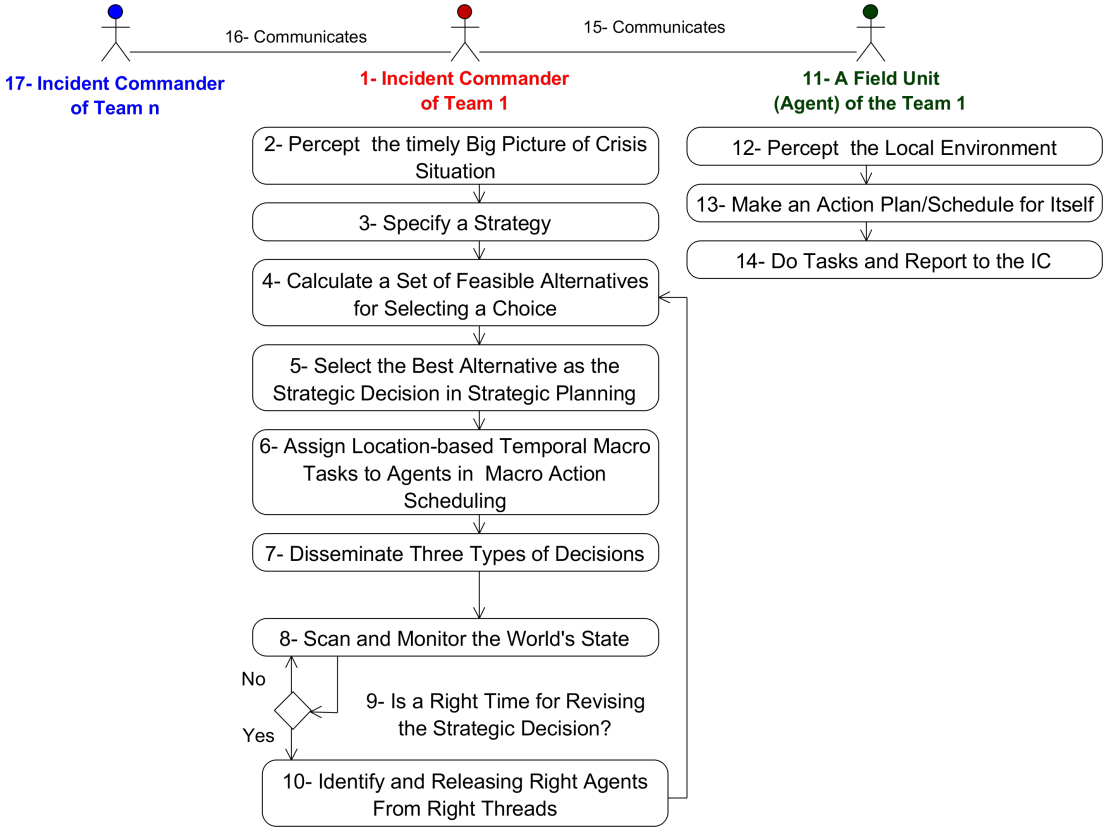


FIGURE 1.2: Work flow of multi-agent coordination in a disaster response team.

1.5.1 The Problem Domain

Review of the past **earthquake disasters** recognizes the importance of efficient emergency response. Urban search and rescue (USAR) are considered as the major part of disaster emergency response operations that their objective is to reduce number of fatalities in the first few days after disaster. A number of different teams and organizations such as Red Crescent Society rapid response teams, INSARAG teams, volunteer teams, fire-fighting teams, medical services, or road-clearing bulldozers are involved in these operations to respond to crisis situations or support activities of responder teams.

The domain of emergency/crisis response is concerned with reducing number of fatalities in the first few days after disaster (natural or human-made), and USAR (Urban Search

and Rescue) has a significant issue in this domain. USAR operations involves four task types: (1) reconnaissance and assessment by collecting information on the extent of earthquake damage, (2) search and locate victims trapped in collapsed structures, (3) extract and rescue trapped victims, and (4) transport and dispatch injured survivors to hospitals and refuges; Rescue tasks, also, are classified into three categories: light, medium, and heavy rescue according to their capabilities requirements.

Teams are often organized hierarchically that this is especially true for command and control organizations in the military and emergency response. A disaster response team such as a firefighting team, Volunteers group, medical emergency services, or INSARAG has a hierarchical structure which is composed of an incident commander in the top level of the team and some agents in the down node of the hierarchy.

Different disasters e.g. flood, fires, landslide, and tsunami disasters include different tasks that should be considered in the problem statement. Although the earthquake disaster response domain may include different types of task, our focus is on the USAR operations. Chapter 3 is dedicated to description of this domain. It is possible to model and present other task types in the problem definition.

1.5.2 Agents

In this paper, a field person or robot is called an agent. In real domains, a team of agents– field units or robots – is faced with the problem of carrying out geographically dispersed tasks under evolving execution circumstances in a manner that achieves a high-level objective in a minimum time. These agents are spatially distributed in a geographic environment, have different capabilities, move from one location to another, perceive their local environment, execute strategic decisions made by their incident commander (IC), have partial information of the world‘s state, make fully decisions about their own actions, coordinate their actions with each other, cooperate with each other, perform various tasks, and report to the operations center; In addition, there is an uncertainty in actions duration and outcome of actions. Role of the IC is to coordinate and control agents because he has a global and big picture of the world's state.

1.5.3 Tasks

Tasks form a local view that an agent percepts from the surrounding environment. Task, which are carried out by agents, are spatially distributed among in the area among geographic objects (building, road segments, city blocks, or zones). Each task may require one capability or several synchronous capabilities, entail various dependencies,

have dynamic and temporal quantities, and take considering time to be completed. Agents have incomplete and uncertain information about tasks (spatial distribution, quantity, and duration of tasks), and new tasks may be revealed by actions of agents in time and space.

1.5.4 Location-based Temporal Macro Tasks

A part of the district 17 of the city of Tehran was chosen to be the case study whose area equals to 0.62 square kilometers. Geographic layers of this area, which are prepared in GIS, were exported to the spatial database.

A LoTeM (Location-based Temporal Macro) task is an aggregative task that aggregates a subset of tasks that has two criteria: 1) identified tasks are from a same task type and 2) theses tasks are spatially contained within a specific geographic object (or adjacent to a road segment) to which this LoTeM task is located. LoTeM tasks are encoded by geographic information. They provide the global picture of the tasks environment for the IC.

In a definite time, a LoTeM task contains two variables: 1) an "enabled" number of tasks and 2) a "not yet enabled" number of tasks. The enabled variable states that how many available tasks are observed, discovered, or revealed within an associated geographic object. Agents can do only tasks which have become enabled. The not yet enabled amount states that how many homogeneous tasks are estimated to be revealed or discovered at future. These two amounts have dynamic and uncertain quantities that may vary over time because some agents accomplish the enabled part while other agents may reveal the another part. These amounts give an estimation of a total duration and total capabilities which are required to do this task type in a geographic area.

There are interdependencies between LoTeM tasks which are associated with a specific location. For example, reconnaissance LoTeM task (enabled amount plus not yet enabled amount) can enable the not yet enabled part of the search LoTeM task within the same geographic area.

1.5.5 Centralized Coordination

In the organization theory [6], strategic management consists of four basic elements: environmental scanning, strategy formulation, strategy implementation, and evaluation and control in order to achieve organizational objectives. Strategic planning is a co-ordination approach for managing tasks relationships by objectives (goal selection and

goal decomposition) and grouping people into units. For example, the incident command system (ICS) is a top-down approach that applies strategic/macro planning for coordination of actions of operational unites. The NIMS ICS system makes an incident action plan in five phases: 1) understand the situation, 2) establish incident objectives (priorities, objectives, strategies, tactics/ tasks), 3) develop an action plan, 4) prepare and disseminate the plan, 5) continually execute, evaluate, and revise the plan.

1.5.6 Assumption

Efficient coordination is challenging and difficult to be achieved because of the complexity of the environment and situations. A complex approach is required to be used in this domain. Therefore, our methodology is to specify a number of sub-challenges that require significant contributions to address the main research question. As a result, in order to achieve this high-level goal, it is important and necessary to identify some key research questions related to development of the ideal system.

- The decision-maker is the IC, therefore coordination is a centralized approach but execution of tasks is distributed among agents.
- Tasks information that forms the IC‘s global perception has the spatial, macro, dynamic, and temporal characteristic; it means that these data should be used in related algorithms.
- Planning, task assignment, and scheduling techniques in multi-agent systems are used for problem solving.
- Methods which are applied should partially constrain agents with macro decisions and permit agents to adapt their activities and make their own tactical (micro) decisions according to real situations.
- Because of the geographic characteristic of the problem, GIS (geographic information systems) are required to support human decisions by providing a set of proper tools for management, analysis, modeling, and visualization of geographic information and location-based information.
- A mixed-initiative system can be proper system for the IC.
- This system does not focus on gathering, integrating, fusing, or mining data and information. So a information system should provide required information for this system.
- IC and field units are equipped to advanced devices that enable them to share information.

1.6 Requirements

This section is dedicated to seven subsection that each one presents a research question and requires a contribution.

1.6.1 Design a GIS-based Assistant Software Agent for the IC

This issue addresses the components 1-10 presented in Fig. 1.2. It aims to design an intelligent software system for the IC. It assists and collaborates with the human in strategic planning and macro tasks assignment for centralized multi-agent coordination [11].

1.6.2 Design a Data Model

This issue addresses the components 2-10 presented in Fig. 1.2. It aims to 1) completely analyze and describe the problem and 2) model, formulate, and present data of the problem completely. A data model presents elements of the problem, properties, relationships, and interaction among these elements with regard to problem data modeling. This data model is important to support development of an appropriate system [10].

1.6.3 Develop Intelligent Algorithms for Assignment of Agents to Human Strategy

A high-level strategy specification enables an I.C. to express and encode his intuition and initiative for multi-agent planning problems solving. A strategy partitions and decomposes a complex problem into a set of small problems under human supervision. A strategy contains a set of parallel, interdependent, and prioritized threads. A thread is a subproblem that is composed of a unique ranking, a subset of agents, a subset of task types, and a subset of geographic area.

This issue addresses the components 4,8,9,10 presented in Fig. 1.2. These algorithms have two key issues: 1) automatically calculate and present a set of feasible alternatives for selecting a choice as a strategic decision in execution of the human strategy in a definite time and 2) autonomously and timely identify a subset of assigned agents that should be released from their threads in order to revise a strategic decision to the new situation [13].

1.6.4 Develop an Automated Algorithm for Assignment of Location-based Temporal Macro Tasks to Agents

This issue addresses the component 6 presented in Fig. 1.2. This algorithm is required to dynamically assign spatial-temporal macro tasks to agents under human strategic decisions in centralized scheduling in order to minimize the overall time of tasks execution. Two main results are achieved by running this algorithm (1) a feasible macro action schedule and (2) an adaption time of the strategic decision [12].

1.6.5 Develop A* Search Algorithm for Optimal Strategic Decision Making in Spatial Multi-agent Planning

This issue addresses the component 5 presented in Fig. 1.2. This algorithm aims to select the best alternative in a strategic decision-making problem in order to minimize the overall time of tasks execution. Results support human decisions and assist the IC to evaluate the quality of his strategy, refine it, or define a better strategy [8].

1.6.6 Simulation of an Organization of Spatial Intelligent Agents

This issue addresses the component 1, 16, 17 presented in Fig. 1.2. Simulation of a society of software agents is an essential problem for implement and evaluation of distribution algorithms for decentralized coordination among these agents. It addresses the problem of development of GIS-based assistant software agents that these agents are embedded in a society and aim to coordinate multiple incident commanders as human planners in planning and scheduling. An agent, with which an incident commander is equipped, is required to 1) provide GIS tools for its human, 2) interact with human, 3) communicate with other agents, 4) manage geographic information, 5) contain problem-solving algorithms, and 6) contain distributed coordination techniques. It is required to develop these social agents and simulate an environment/society of these agents [14].

1.6.7 Simulation of Intelligent Mobile GIS

This issue addresses the components 11,12, 13 presented in Fig. 1.2. It aims to simulate a society of distributed intelligent mobile GIS. A human as a field unit is equipped with a SpDI2A and forms a human-agent team. Human-agent teams cooperate with each other in order to optimize coordination of field units by distributing and allocating tasks among them in time and space efficiently and sufficiently. [9].

1.7 Conclusion

This chapter analyzed the coordination problem within a disaster response team from the IC's point of view. It decomposed this complex problem into subproblems and designed a number of requirements for development of spatial intelligent coordinators for incident commanders.

This chapter proposed seven key research questions that address research questions as seven research challenges which are considerable importance in developing an ideal solution for the addressed problem. This framework presents a road map that states what necessary sub-issues/contributions we should achieve in order to develop GIS-based intelligent assistant agents. It specified a research plan for future works.

The future work is to extend the designed requirements and specify new sub-problems that should be addressed. Three ideas are considerable for future works in this domain: 1) develop distributed intelligent mobile GIS for filed units for distribution of tasks among themselves, 2) simulate a spatial agent-based model of teamwork in a disaster responder team, and 3) allocate refuges to location-based tasks under a human strategic decision.

References

[1] Balasubramanian, Vidhya, Daniel Massaguer, Sharad Mehrotra, and Nalini Venkatasubramanian. Drillsim: A simulation framework for emergency response drills. Springer Berlin Heidelberg, 2006.

[2] Barbulescu, Laura, Zachary B. Rubinstein, Stephen F. Smith, and Terry L. Zimmerman. "Distributed coordination of mobile agent teams: the advantage of planning ahead." In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, pp. 1331-1338. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[3] Berry, Dave, Asif Usmani, Jose L. Torero, Austin Tate, Stephen McLaughlin, Stephen Potter, Arthur Trew, Rob Baxter, Mark Bull, and Malcolm Atkinson. "FireGrid: Integrated emergency response and fire safety engineering for the future built environment." UK e-Science Programme All Hands Meeting, 2005.

[4] Bigley, Gregory A., and Karlene H. Roberts. "The incident command system: High-reliability organizing for complex and volatile task environments." Academy of Management Journal 44, no. 6 (2001): 1281-1299.

[5] Jennings, Nick, Sarvapali D. Ramchurn, Mair Allen-Williams, Raj Dash, Partha Dutta, Alex Rogers, and Ioannis Vetsikas. "The ALADDIN project: Agent technology to the rescue." In Proceedings of the First Intl. Workshop on Agent Technology for Disaster Management. 2006.

[6] Hunger, J. David, and Thomas L. Wheelen. Essentials of strategic management. New Jersey: Prentice Hall, 2003.

[7] Kitano, Hiroaki, and Satoshi Tadokoro. "Robocup rescue: A grand challenge for multiagent and intelligent systems." AI Magazine 22, no. 1 (2001): 39.

[8] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, and Stephen F. Smith, "A* Search Algorithm for Optimal Strategic Decision Making in Spatial Multi-agent Planning." International Journal of Geographical Information Science, 2014. Under review.

[9] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[10] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research, 2013. (in press)

[11] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." In Workshop on Robots and Sensors integration in future rescue INformation system, ROSIN'13. In conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), 2013.

[12] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC). (in press)

[13] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, and Pedro Szekely. "Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination." Journal of Software, 2014. (under review)

[14] Nourjou, Reza, and Michinori Hatayama. "Simulation of an Organization of GIS-based Assistant Software Agents in C#.Net Framework.", International Journal of Computer Theory and Engineering , 2014. (under review)

[15] Schoenharl, Timothy, Greg Madey, Gábor Szabó, and Albert-László Barabási. "WIPER: A multi-agent system for emergency response." In Proceedings of the 3rd International ISCRAM Conference, pp. 1-7. 2006.

[16] Schurr, Nathan, Janusz Marecki, John P. Lewis, Milind Tambe, and Paul Scerri. "The defacto system: Training tool for incident commanders." In AAAI, pp. 1555-1562. 2005.

[17] Vafaeinezhad, Ali Reza, Ali Asghar Alesheikh, Majid Hamrah, Reza Nourjou, and Rouzbeh Shad. "Using GIS to Develop an Efficient Spatio-temporal Task Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications–ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009.

Chapter 2

Design a GIS-based Assistant Software Agent for Incident Commander

2.1 Abstract

Problem: This chapter addresses the design of an intelligent software system for the IC (incident commander) of a team in order to coordinate actions of agents– field units or robots –in the domain of emergency/crisis response operations.

Objective: This chapter proposes GICoordinator (Gis-based Intelligent Coordinator). It is a GIS-based assistant software agent that assists and collaborates with the human planner in strategic planning and macro tasks assignment for centralized multi-agent coordination.

Method: Our approach to design GICoordinator was to: analyze the problem, design a complete data model, design an architecture of GICoordinator, specify required capabilities of human and system in coordination problem solving, specify development tools, and deploy.

Result: The result was an architecture/design of GICoordinator that contains system requirements.

Findings: GICoordinator efficiently integrates geo-informatics with artifice intelligent techniques in order to provide a spatial intelligent coordinator system for an IC to efficiently coordinate and control agents by making macro/strategic decisions. Results define a framework for future works to develop this system.

2.2 Introduction

The domain of emergency/crisis response is concerned with reducing number of fatalities in the first few days after disaster (natural or human-made), and USAR (Urban Search and Rescue) has a significant issue in this domain. A disaster response team, which contains an IC and several agents, is faced with the problem of carrying out geographically dispersed tasks under evolving execution circumstances in a manner that achieves a high-level objective in a minimum time. Agents need to efficiently coordinate their actions with each other in order to maximize the objective function. Effective coordination is an essential ingredient for efficient emergency response management but it is difficult to achieve.

It is important for the IC, who has a big picture of the state of the world, to coordinate and control agents. His main role is to make a strategic action plan, allocate tasks to agents, decide on actions of agents, and schedule activities of agents in time and space.

To propose an ideal system, some requirements should be considered as follows. The decision-maker is the IC, therefore coordination is a centralized approach but execution of tasks is distributed among agents. Tasks information that forms the IC‘s global perception has the spatial, macro, dynamic, and temporal characteristic; it means that these data should be used in related algorithms. Planning, task assignment, and scheduling techniques in multi-agent systems are used for problem solving. Methods which are applied should partially constrain agents with macro decisions and permit agents to adapt their activities and make their own tactical (micro) decisions according to real situations. Because of the geographic characteristic of the problem, GIS (geographic information systems) are required to support human decisions by providing a set of proper tools for management, analysis, modeling, and visualization of geographic information and location-based information. A mixed-initiative system can be proper system for the IC.

This requires an ideal intelligent software system to assist the IC and collaborate with him in strategic planning and tasks assignment to agents according to the assessed requirements.

Although there is much literature [4], [2], [1] in planning, coordination, task assignment, human-machine collaboration, problem-solving algorithms, and decision making under uncertainty, unfortunately, the discussed requirements have not been thoroughly addressed by the previous works.

Designing an ideal approach is an important phase in system development. As a result, in order to develop an ideal system, this chapter aims to design GICoordinator. GICoordinator is a GIS-based assistant software agent that assists and collaborates with the

human planner in strategic planning and macro tasks assignment in the mixed-initiative approach, which was dissussed in Section 1.5. This chapter summarizes phases that make up this system.

2.3 System Design

This section is dedicated to phases which are important to design the GICoordinator.

2.3.1 Analyze the Problem

First step is to completely analyze and describe the problem stated in Fig. 1.2. Essential dimensions that compose this problem include: the problem domain, the structure of a team, geographic information, macro tasks, requirements, assumptions, the goal function, the strategic action plan, and the macro task schedule [3].

2.3.2 Design the Architecture of GICoordinator

The architecture of GICoordinator is designed based on integration of three key components as Fig. 2.1 shows. According to Fig. 1.2, Table 2.1 defines the required functionalities of GICoordinator and human according to the analyzed problem. The following subsections briefly describe some important capabilities.

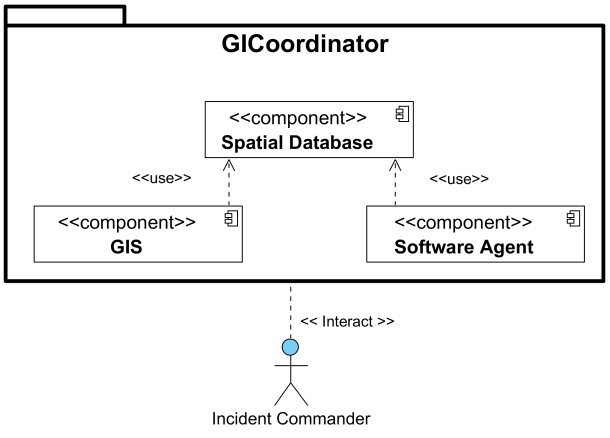


FIGURE 2.1: The architecture of GICoordinator

TABLE 2.1: System Requirements: the Desirable Capabilities of GICoordinator and Human with regard to the problem stated in Fig. 1.2

Component	**Functionalities**
Human	1- Specify high-level strategies for coordination of agents 2- Revise and refine strategies
Spatial Database	3- Organize data (geographic, location-based, and non-geographic) of the problem 4- Integrate the software agent with GIS 5- Support information sharing with agents and with other information systems, and support integration of information systems
GIS	6- Support human decisions by providing tools for geospatial reasoning, geographic information management and visualization 7- Support the software agent by providing GIS analysis 8- Interact with human
Software Agent	9- Assign agents (field units) to human high-level strategies in strategic planning 10- Assign geospatial-temporal macro tasks to agents in centralized scheduling 11- Adapt and revise the strategic decision 12- Adapt and revise the schedule 13- Search for an optimal plan 14- Allocate resources (refuges) to damage points under human strategies 15- Adjust and refine human strategies 16- Provide an interface to interact with human 17- Percept and observe the environment via the spatial database

2.3.2.1 Provide GIS Analysis

GICoordinator provides geo-technologies and geo-informatics that support human decisions and increase functionality of the software agent. Three significant purposes are of importance: (1) visualization of information via 3D maps or thematic maps e.g. display spatial distribution of tasks, states of macro tasks, action plans, and tasks schedule, (2) geospatial reasoning that includes spatial relationships analysis, network analysis, proximity analysis, etc, and (3) information management that includes retrieval, insert, query, update, and manipulate information of spatial database. It enables the IC to interact with geographic information, and it supports human decisions by providing a better situational awareness.

GICoordinator requires necessary information for computation because it is mainly focus on multi-agent coordination techniques. Different types of information are provided by distributed systems. Therefore GICoordinator requires another capability to gather, integrate, mine, or fuse data from distributed databases. This version of GICoordinator does not include this functionality.

2.3.2.2 Strategic Planning

It is an approach in an organizational structure to make a strategic plan that states that how agents can get from the current state of the world through a sequence of actions to a desired goal state. Strategic planning in a disaster response team includes (1) specify a response objective (a high-level strategy) for the team and decompose it into prioritized sub-goals (threads), (2) make a strategic decision by assignment of agents to threads, and (3) evaluate and adapt a strategic decision to new crisis situations.

This capability calculates a set of right choices for making a strategic decision either in execution of the human strategy or in adaption of current assignments. These choices are presented for the IC to select the best choice according to his intuition or to delegate the system to search for the optimal one. Moreover, the system autonomously releases right agents from a right thread in a right time during tasks execution in order to revise the strategic decision.

2.3.2.3 Centralized Scheduling

This capability is required to dynamically assign spatial-temporal macro tasks to agents under human strategic decisions in centralized scheduling in order to minimize the overall time of tasks execution. Two main results are achieved by running this algorithm: (1) a feasible schedule and (2) an adaption time. A schedule is composed of a number of macro decisions that specify: (1) what task type is going to be done, (2) who (a subset of agents) are assigned to do this assignment, (3) where (a macro geographic object) contains a subset of tasks, (4) when operations start, (5) when operations finish, (6) how many tasks are estimated to be done, and (7) what task types and how many of them are estimated to be revealed in this location after to finish this job.

2.3.2.4 State-Space Search

This capability calculates an optimal strategic plan, a complete schedule, and a overall minimum time of tasks execution. It evolves assignment of agents to threads from an

initial state to a specified goal state. Results state that how and when agents can reach a defined objective. These results support human decisions and assist the IC to evaluate the quality of his strategy, refine it, or define a better strategy.

2.3.2.5 Resource Allocation

This capability optimally allocates available refuges to rescued people in medical transportation operations according to human strategic decisions. Results state that which injured persons should be transported to which refuges in order to optimize an objective function.

2.3.2.6 Adjustment of Human Strategies

It is difficult for an IC to specify a good strategy and timely revise it during emergency management. An IC may specify a bad or wrong high-level strategy that leads to a very big catastrophe with significantly severe consequences. This capability recommends human for adjustment and refinement of his strategy in real-time.

System should resolve trade-off of resource/role assignments and reflect feed back from actual damaged area. This system requirement should consider them. Machine learning algorithms can provide proper solution. We will devote a chapter to this functionality.

2.3.3 Design the Data Model

It is important to model, formulate, and present data of the problem completely. The data model presents elements of this problem, properties, relationships, and interaction among these elements with regard to problem data modeling. This data model is important to support development of GICoordinator and implementation of the capabilities designed for this system [3].

The designed data model only formulates the planning & scheduling problem which the IC is faced, although spatially distributed agents can observe, gather, and report many types of information from their local environment to the incident center. Uncertainties are presented by proper attributes of classes of this data model, and they are used by related algorithms.

This data model formulates and present any USAR operations and similar crisis operations. It, also, formulates a team of different agents. The size of a team and the size of disaster scenario are scalable, but it is important to model any required information in the data model.

TABLE 2.2: Methodology for Development of the GICoordinator

Component	**Tools**
Spatial Database	Microsoft Spatial SQL Server, GeoDatabase, the data model
GIS	ArcGIS, .NET Programming
Software Agent	C#.NET Programming, ArcObjects, the data model

2.3.4 Development tools

There are many tools and approaches that can be used to implement and develop a system. In our methodology, Table 2.2 shows required tools that were used to develop GICoordinator. It states that how each component should be implemented and developed.

We applied the C#.Net programming language for developing GICoordinator. Algorithms, the structure of the system, and rules were implemented in the program. All system requirements, which are defined in Table 2.1, were implemented by the developer using the specified development tools.

There is a key concern in development of GICoordinator. The designed data model was used to develop his system, and there are interdependencies among functions of the system. All functions should be run in an integrated system. Any change in the data model may cause that the system does not work.

2.3.5 Deploy

GICoordinator is deployed in two ways. First one is to calculate an feasible strategic plan & schedule, which partially specify actions of agents, by the human-system collaboration before execution. Second one is to use this system for automated and autonomous adaption/refinement of macro/strategic decisions to new situations during execution and in real time.

As Fig. 2.1 shows, the GIS and the software agent have a connection to the spatial database. In oder to run GICoordinator, the spatial database, whose structure is based on the designed data model, has to contain essential information of the initial state of the world. These data can insert or modified by the human via the user-interface that GICoordinator provides for human-system interaction [3].

2.4 Conclusion

The design of GICoordinator was discussed in this chapter. The key insight is (1) support human decisions with geo-spatial intelligent software system, (2) provide A.I. techniques for strategic planning, macro tasks assignment, scheduling, and automated adaption of these decisions in central multi-agent coordination, (3) involve human in the loop and enable collaboration between human and system for decision making. Future works will be to address the defined capabilities of GICoordinator by several papers.

References

[1] Maheswaran, Rajiv T., Pedro Szekely, and Romeo Sanchez. "Automated adaptation of strategic guidance in multiagent coordination." In Agents in Principle, Agents in Practice, pp. 247-262. Springer Berlin Heidelberg, 2011.

[2] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[3] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research. (submitted)

[4] Vafaeinezhad, Ali Reza, Ali Asghar Alesheikh, Majid Hamrah, Reza Nourjou, and Rouzbeh Shad. "Using GIS to Develop an Efficient Spatio-temporal Task Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications–ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009.

Chapter 3

Data Model for Strategic Planning and Scheduling

3.1 Abstract

Problem: Strategic action planning and scheduling (SAP) in coordination of a disaster response team includes objective selection and decomposing it into sub-goals, grouping available units into coalitions and assigning them to the sub-goals, allocation of the units to tasks, and adaption of the made decisions. SAP is the major responsibility of the incident commander of a team for coordination of operational units actions during disaster crisis/emergency response management by making macro/strategic decisions.

Objective: This chapter intends to completely model, formulate, and present data of the SAP problem. This data model is used to support design and development of an appropriate approach for SAP.

Method: The methodology was to analyze and study the SAP problem. A SAP problem is composed of six essential dimensions: the problem domain, geographic information, geospatial-temporal macro tasks, strategic action planning, strategic action scheduling, and the structure of a team.

Result: The contribution of this chapter is the SAP problem data model. It is designed as an UML class diagram consisting of entity types, attributes, and relationships with regard to SAP problem data modeling.

Conclusion: To evaluate the quality of SAP data modeling, the SAP problem data model was used in proposing and developing an intelligent assistant software system that assists and collaborates with an incident commander for SAP. Results shown five

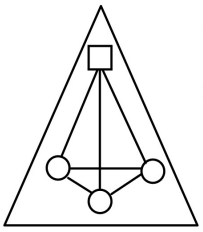


FIGURE 3.1: Organizational structure of a disaster response team.

new findings: 1) a complete data model of the SAP problem, 2) present and summarize tasks information in geographic objects, 3) express and encode humans' intuition as a human high-level strategy guidance for SAP, 4) formulate a strategic action plan, and 5) integrate strategic action schedules information with other entities.

3.2 Introduction

Review of the past disasters recognizes the importance of efficient emergency response. Urban search and rescue (USAR) are considered as the major part of disaster emergency response operations that their objective is to reduce number of fatalities in the first few days after disaster [10]. A number of different teams and organizations such as Red Crescent Society rapid response teams, INSARAG teams, volunteer teams, fire-fighting teams, medical services, or road-clearing bulldozers are involved in these operations to respond to crisis situations or support activities of responder teams.

A disaster response team is a hierarchical organization because its structure essentially consists of two levels. The down level or the operational level includes a number of different field units who are rational and semi-autonomous agents. Their roles are to perceive their local environment, follow and execute decisions made by the top level of the team, make their own decisions, coordination their actions with other units, accomplish tasks according their plan, and report to the incident commander post on their local observations. An incident commander is located at the top node in the hierarchy and has a global view of the enviroment. His or her important role as a human planner is to make action plans and schedules for field units. Therefore, a team has an organizational structure composed of cooperative agents distributed in different levels. **Fig. 3.1** shows the structure of a team and characteristics of the two levels.

Effective coordination is a crucial issue for emergency response management [6]. Coordination is difficult and hard because of characteristics which this domain includes such as: uncertainty in information, time, limited resources, task flow uncertainty, etc [5]. The coordination theory states that coordination is the act of managing interdependencies among activities performed to achieve a goal [23]. Coordination in disaster

emergency response includes management of task flow (tasks and interdependent relationships), recourse, information, decision, and responder [6]. Inefficient coordination results in "idle" units (gents), conflict between actions, or "redundant" activities that operations take a very long duration to be completed.

Planning and scheduling provide two major mechanisms of coordination for managing task dependencies and shared resources [6], [5],[23]. Crisis response systems should utilize these mechanisms in formulating crisis management [14], [17]. An approach to coordination in agent-based systems is to engage the agents in multi-agent planning and scheduling [34]. The problem of how agents should get from the current state of the world through a sequence of actions (a plan) to the desired goal state states a planning problem in multi-agent systems. The scheduling problem in multiagent systems is the problem of suitable assignment of limited resources (agents) to time-consuming tasks within a specified time window and coping with a set of constraints over time in order to maximize an optimization criterion.

In the organization theory, strategic management consists of four basic elements: environmental scanning, strategy formulation, strategy implementation, and evaluation and control in order to achieve organizational objectives [12]. Strategic planning is a coordination approach for managing tasks relationships by objectives (goal selection and goal decomposition) and grouping people into units [23]. For example, the incident command system (ICS) is a top-down approach that applies strategic/macro planning for coordination of actions of operational unites. The NIMS' ICS system makes an incident action plan in five phases: 1) understand the situation, 2) establish incident objectives (priorities, objectives, strategies, tactics/tasks), 3) develop an action plan, 4) prepare and disseminate the plan, 5) continually execute, evaluate, and revise the plan [3],[1],[9]. Strategic action planning and scheduling (SAP) is a cordination mechanism (management of tasks flow and shared resources) within a team. SAP includes objective selection and decomposing it into sub-goals, grouping available units into coalitions and assigning them to the sub-goals, allocation of the units to tasks, and adaption of the made decisions. In another word, to solve a SAP problem results in three types of decisions: a high-level strategy, a strategic action plan, and a strategic action schedule. SAP versus tactical decision-making, which is done by units, makes macro decisions for activities of the team that constrain and limit micro activities of field units in the down level. SAP, therefore, is a critical issue in a team. **Fig. 3.2** shows The workflow of SAP which is done in the top level of a team. The SAP problem will be discussed in Section 3.

SAP is the key task of incident commanders as human planners of teams for coordination of operational units' actions during disaster emergency response management. As a result, it is important and necessary to develop any appropriate approach for SAP. However, several approaches which most of them are based on the multiagent systems have been developed for coordination of disaster/crisis emergency response [1],[36],[29],[11],

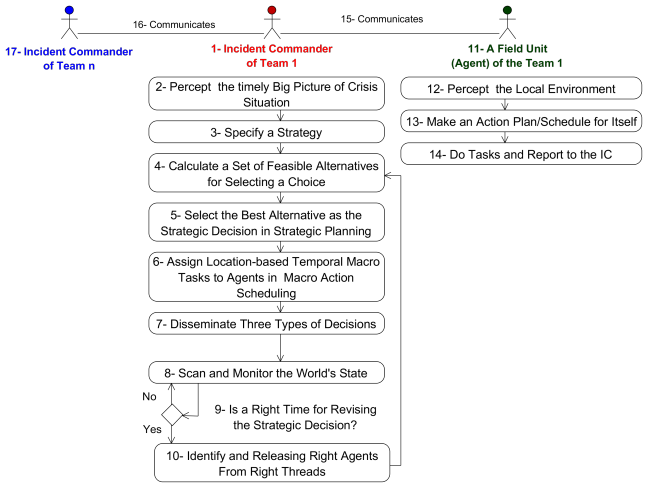


FIGURE 3.2: The work flow of strategic action planning and scheduling in a team.

[18],[7],[2],[35], [22],[15], unfortunately they do not provide appropriate solutions for SAP.

According to SAP, limitation and caps identified in the related works are classified into four categories.

1. Automated planning systems vs. human-machine collaboration. Automated planning systems do not let human planners be involved in SAP. These systems are not mixed with human's intuition and do not collaborate with human for SAP. The main obstacle is scale as it is currently infeasible for a fully automated system to effectively reason about all the possible futures that may arise during execution of tasks in a complex environment [21]. Mixed-initiative planning systems are systems in which humans and machine collaborate in the development and management of plans by providing capabilities each one does best [4]. In general, they can often produce better solutions for complex problems.

2. Tactical decisions vs. strategic ones. Tactical decisions (action plan and schedule) concern the down level and exactly specify micro actions of agents (units). But, it is impossible for incident commanders to fully specify this type of decisions for agents.

3. Micro tasks information vs. macro one. These systems uses micro tasks for make a global view of the SAP problem while the top level does not access to micro tasks information. Operations centers gather information and data from different resources and integrate them to make a global picture of the environment. Thus, incident commanders have inaccessible, global, and uncertain information of the state of whole tasks environment, but agents have direct, complete, and accurate information about their local environment's state.

4. Non-geographic information vs. geospatial one. Related works are not integrated with geographic information systems (GIS) and neglect the importance of geographic information in SAP. Because the SAP problem has the geospatial aspect, it requires GIS. GIS provide sufficient tools for incident commanders to analyze, visualize, and manage geographic and location-based information [16], [11].

With regard to the limitations addressed in previous works, it is required to propose, develop, apply any appropriate approach (intelligent software system) for SAP. An ideal system collaborates with human (incident commander) to make strategic decisions with regard to macro tasks information and geographic information. The first critical phase to achieve this goal is to model data of the SAP problem.
SPA problem data modeling is the objective of this chapter. It is important and necessary to completely model, formulate, and present data of the SAP problem in order to support development of any appropriate approach for SAP. The data model is one of the most critical tasks in the entire systems development process [26]. The data model is a major determinant of system development costs, system flexibility, integration with other systems and the ability of the system to meet user requirements. A data model presents elements of a problem, their properties, and relationship and interaction among these elements. Two research questions arise in this chapter. What is the SAP problem? What is the SAP problem data model?
The contribution of this chapter is the SAP problem data model. It is designed as an UML class diagram consisting of entity types, attributes, and relationships with regard to SAP problem data modeling. This data model was used for three purposes: 1) implement a geo-database, 2) design and develop an assistant software agent (GICoordinator), and 3) implement a geographic information system. Five main characteristics which our contribution includes are as follows.

1. The SAP problem data model. It models and formulates the SAP problem. This data model is important and essential to implement desirable features of an assistant software agent for SAP successfully.

2. Geospatial-temporal macro (GTM) tasks information. The SAP data model presents and summarizes tasks information in different geographic objects (buildings, city blocks, etc) and extract tasks information from one geographic layer for another layer regarding spatial topology relationships between layers. A geospatial-temporal macro task is the accumulation of all tasks (enabled tasks and disenabled tasks) with a same task type that they are spatially contained within the geographic object of this macro task in a specific time.

3. Human high-level strategy guidance. The SAP problem data model formulates and encodes humans' intuition as a human high-level strategy guidance for SAP. It enables human planner to express and specify their intuition for the intelligent assistant system to make strategic action plans and schedules in a collaborate approach.

4. Strategic action plan. This data model formulates a strategic action plan and integrate with other entities. To execute the strategy, i.e., make optimal assignments/allocations of agents to threads results in a strategic plant which is presented by this data model.

5. Strategic action schedule. The SAP data model integrates and presents temporal assignments (allocation) of agents to GTM tasks. An instance of a strategic action schedule determines: 1) location of a macro task assigned, 2) the task type of the macro task, 3) start time, 4) finish time, 5) amount of done tasks, and 6) a set of agents allocated to this GTM task.

3.3 Review the Related Works

3.3.1 Comparison Criteria

Different data models have been developed to address different problems. Each one aims to address a number of characteristics & aspects that an approach requires in solving a specific problem. It is obvious that various problems requires various data modeling according to their demands and requirements. In fact, complexity and efficiency of a data model is dependent to a characteristic/requirements of a problem. For an example, number of teams, level of hierarchy, recourses management, role of filed units in planning and scheduling, complex structure of tasks, etc affect formulation and presentation of a problem.

With regard to the importance of SAP, this chapter discussed the SAP problem and proposed the SAP data model for formulating and modeling this problem. Their completeness are compared according the the six characteristics: 1) geographic information, 2) macro tasks environments, 3) human-machine collaboration, 4) strategic action planning, 5) strategic action scheduling, 6) and the IC (top level of the team).

The question is that whether there is a previously developed data model that can completely model this problem? A perfect data model should consider six main characteristics: 1) the decision-maker is the IC, who is located in the top level of a team, 2) geographic and location based information should be modeled, 3) tasks have spatial,

TABLE 3.1: Features of the related works with regard to SAP data modeling.

Related works	Features					
	1	***2***	***3***	***4***	***5***	***6***
[29]	B	A	B	B	B	-
[36]	A	A	A	B	-	B
[18]	A	A	B	B	B	B
[2]	B	B	B	B	B	B
[22]	A	B	B	A	A	B
[11]	A	A	A	C	-	-
[35]	A	B	B	A	-	-

Features Description:
1: Decision-Maker Level. A- Top (IC), B- Down
2: Information. A- Geographic, B- Non-Geographic **3**: Tasks Scale. A- Macro, B- Micro
4: Approach. A- Human-Machine Collaboration, B- Automated, C- Human
5: Plan Type. A- Strategic, B- Tactical
6: Schedule Type. A- Strategic, B- Tactical

macro, and temporal characteristics, 4) human intuition should be mixed/involved in the problem-solving techniques, 5) a plan includes strategic/macro decisions, and 6) a schedule includes strategic/macro decisions.

This comparison will show the originality of the SAP data model.

3.3.2 Review

This section reviews and compare some related works and their deficiencies and limitations with regard to SAP data modeling. **Table 3.1** summarizes the features of these works according to the SAP problem data modeling.

A simple conceptual model of spatial coordination in an USAR team was designed and used in development of spatially distributed intelligent assistant agents and geo-simulation of an USAR scenario [29]. In this data model, a damaged building contains a group of search and rescue tasks that each search task can release or discover a rescue task. Each task should be assigned to one proper field unit. This data model focuses on distributed coordination among agents by allocating and distributing micro tasks. This approach is not applicable for SAP.

A conceptual model was designed by [36] for human group task allocation in rescue management. It was used to implement a greedy spatio-temporal task allocation algorithm in GIS environment. This model contains spatial and non spatial layers (parcels, networks, damage points, tasks, field personnel, task-list, distributed tasks, and cost). This data model is applied by an automated information system that assigns segment-based

tasks to field units. However this approach contain macro tasks, unfortunately it does not consider interdependencies among tasks and each macro tasks can be assigned to only one unit.

The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams acting in large urban disasters [18]. Rescue agents try to minimize damages caused by earthquake such as civilians buried, buildings on fire and blocked roads. In this project, there are the three different teams and six types of agents: 1) a fire station with several fire brigades, 2) a police station with several police forces, and 3) an ambulance center with several ambulance teams. Although this approach contains an IC for each team, but the made decisions are tactical/micro and tasks presented are micro. It, also, does not involve human in the planning loop, and each field agent has a specific capability.

The C-TAEMS [2] modeling language, which is an adaptation of the TAEMS [7], formulates and models distributed multi-agent coordination problems. It was used for development of the COORDINATORS program by different approaches [**?**],[20]. A C-TAEMS problem instance contains a set of agents and a hierarchically decomposed task structure. Each agent has a set of activities known as methods that they can perform. Nodes in the graph are either complex tasks, which each one is composed of a group of tasks and/or methods, or executable methods as leaf nodes which are executed once by a specified agent. Each node may have temporal constraints on the earliest start time and the deadline and non-local effect dependencies that represent hard (enables and disables) and soft (facilitates and hinders) relationships. Methods have probabilistic outcomes for duration, quality, and cost.
STaC adapted the C-TAEMS to involve human's intuition in multiagent coordination [22]. It integrated human strategy guidance with the C-TAEMS to enable human-agent collaboration to improve efficiency of the COORDINATOR project. Unfortunately this approach formulates location-based and micro tasks, and thus tasks assignments made by this approach have the tactical feature. Geographic Information Systems are used for integrating, analyzing and visualizing geospatial data. Emergency management uses geo-information technologies in all five phases of the emergency management process, i.e. planning, mitigation, preparedness, response, and recovery [9]. For example, three collaborative geo-information platforms were developed by to 1) allow synchronous and asynchronous collaboration between decision makers, 2) support GIS use by mobile emergency management teams, and 3) provide open standards-based web portal technologies [11]. Although GIS is used by incident commander systems, they are not enough for SAP. They are required to be integrated with other systems for strategic action planning and scheduling.
DEFACTO incorporates state of the art artificial intelligence, 3D visualization and human-interaction reasoning into a unique high fidelity system for training incident

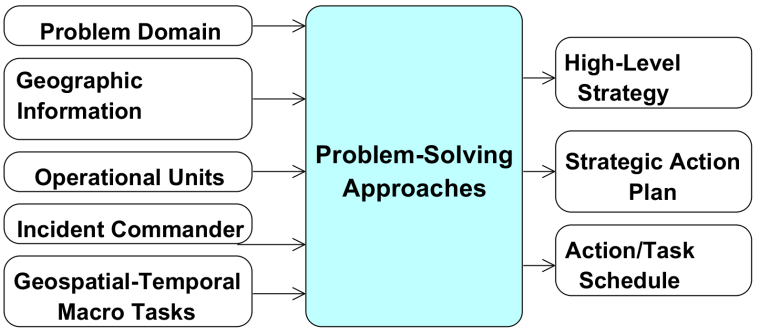


FIGURE 3.3: The SAP problem.

commanders [35]. Key aspect focuses on adjustable autonomy that refers to an agent's ability to dynamically change its own autonomy, possibly to transfer control over a decision to a human or another agent. DEFACTO is comprised of various transfer-of-control strategies. Unfortunately this approach is not related to SAP.
ICS is a disaster management tool based on a series of rational bureaucratic principles for disaster responses. It provides a set of rules and practices to guide the actions of the various organizations responding to disaster, and creates the necessary division of labor and coordination mechanisms among them [3]. Basic system objectives and plans are established at or near the top of the hierarchy and used as bases for decisions and behaviors at lower levels. The incident commander assesses the situation, identifies contingencies, develops objectives, ascertains resource needs, and generates an initial action plan [1]. Unfortunately, FEMA provides a set of useful guidelines about practices [9] but it does not make explicit the design requirements for information systems to make incident action plan.

3.4 Analyze the SAP Problem

The SAP problem is composed of a number of different dimensions. **Fig. 3.3** shows the conceptual model of components that form the SAP problem. This section analyze and describe these dimensions.

3.4.1 The Problem Domain

USAR is the problem domain chose by this chapter because of its major role in earthquake disaster response. The global goal of USAR operation is to rescue the greatest number of people who are trapped under the debris of damaged buildings in the shortest amount of time.

USAR tasks involves a sequence of dependent tasks: (1) reconnaissance and assessment by collecting information on the extent of damage; (2) search and locate victims trapped in collapsed structures; and (3) extract and rescue trapped victims; and (4) transport/dispatch injured survivors to hospitals or refuges. Rescue tasks, also, themselves are classifies into three categories: light rescue, medium rescue, and heavy rescue. In addition, there are other supporting tasks such as road-clearing tasks and fire-fighting tasks that facilitate and support USAR tasks.
To save a person, it is necessary to define a set of the USAR tasks. Sometimes several persons are trapped by a destroyed building. USAR tasks are location-based entities that are distributed in an extensive geographical area.
To accomplish each task needs a considering duration and a specific capability or several synchronous capabilities. Capability requirements determine what agents are allowed to to do what tasks.
In this domain, coordination is managing task flow. The "Enabling" dependency between tasks specifies that when a task is done completely, it makes possibility of performing another dependent task. In other words, time that a disenabled task gets enabled is dependent to the finishing time of a task that makes it enabled or released. For example, the rescue of a trapped person is dependent to the search of that person.

3.4.2 Disaster Response Teams

A variety of responsible or supporting teams are involved in disaster emergency response and crisis management such as Red Crescent Society rapid response teams, INSARAG teams [13], volunteer teams, fire-fighting teams, medical services, or road-clearing bulldozers. **Fig. 3.1** shows the organizational structure of a team. A team is essentially composed of an IC in the top level of the team and several agents (field units) in the down level. They cooperate with each other to achieve objectives of the team.

3.4.2.1 Operational units (agents)

Operational or field units/personals of the team are considered geospatial, mobile, and semiautonomous agents that are distributed in a geographic area. Their main role is to do tasks using their capabilities in the operational area.
Agents may possess heterogeneous capabilities that allow them to engage in doing tasks for which they can provide required capability. Moreover, agents execute their capabilities with different speed. They are categorized according their capabilities and performance into several agent types: (1) "Reconnaissance", (2) "Canine Search", (3) "Electronic Search", (4) "light Rescue", (5) "Medium Rescue", (6) "Heavy Rescue", (7)

volunteer etc.
Agents are required to coordinate their actions with each other for three reasons. First reason is to manage interdependencies among actions of agents because of dependency relationships between tasks. Second one is to manage redundant actions for doing joint tasks. Third reason is to manage agents as shared resources which are assigned to time-consuming tasks. Efficient coordination minimizes the operation time in which all tasks are accomplished.

3.4.2.2 Incident commander (IC)

An Incident commander is a human planner who is located at the top node in the hierarchy of the team. His or her main role is to plan and schedule actions of agents for coordination of disaster response management. **Fig. 3.2** shows the activity diagram of an IC in a team according to SAP.
The IC has inaccessible, global, and uncertain information about the environment's state. His perception/observation of disaster situation is global that is different with agents' perception of their local environment.

3.4.3 Geographic Information

The problem domain is done in a geographical environment which includes different geographic layers such as buildings, city blocks, road network, etc. Each layer is composed of geographic objects. These layers provide base layers to which to geo-locate tasks information, agents, strategic action plan and schedule information.
There are topological relationships between spatial objects of geographic layers [8] e.g. each building is contained within a specific city block while that building is adjacent to a certain road segment.

3.4.4 Geospatial-Temporal Macro Tasks (GTM Tasks)

Macro tasks information forms the global view/perception of ICs from the tasks environment. A macro task is the accumulation of all tasks (enabled tasks and disenabled tasks) that are from a same task type and are spatially contained within a specific geographic object in a definite time. Topological relationships between geographic objects enable the ICs to extract and present macro tasks information for different geographic layers.
A macro task indicates the total number of capability requirement for doing a set of

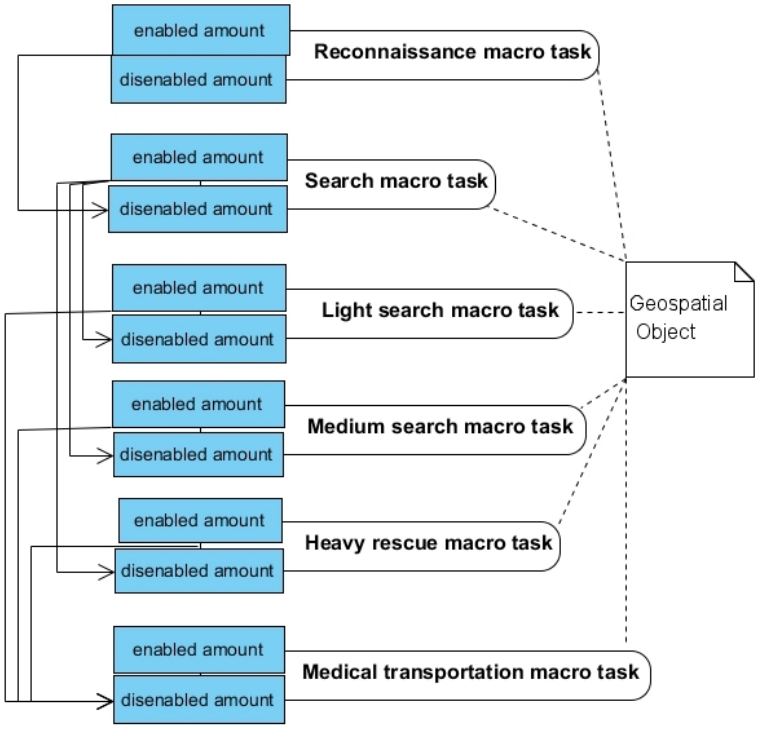


FIGURE 3.4: The task flow diagram of geospatial-temporal macro tasks of a geographic object in a definite time.

homogenous tasks. It gives an estimation of number of required teams and an estimation of operation duration. Because of the temporal environment, the enabled amount and the disenabled mount of macro tasks vary over time. It leads to a series of discrete temporal macro tasks.

Four sources generate tasks information: 1) estimate and forecast, 2) observe and gather tasks data directly, 3) information shared by other teams, and 4) fuse and integrate information.

Macro tasks have the "enabling" dependency among themselves in the USAR problem domain. **Fig. 3.4** shows a task flow of six GTM tasks which are defined for a geographic object. It is clear for e.g. both "Reconnaissance" enabled tasks and "Reconnaissance" disenabled tasks can release and discover "Search" tasks which are disenabled.

3.4.5 Strategic Action Planning

The goal of SAP is to coordinate agents of a team by a strategic action plan which is made by the IC of the team using global vision.

The concept of SAP proposed by this chapter is close to incident action planning process [9], [12]. An incident action plan (IAP) It is built by an incident command system on five phases: (1) Understand the situation; (2) Establish incident/response objectives (priorities, objectives, strategies, tactics, tasks, and work assignments); (3) Develop the

plan; (4) Prepare and disseminate the plan; (5) Execute, evaluate, and revise the plan. SAP assigns/allocates a subset of agents to a subset of GTM tasks. SAP includes two phases: (1) to specify human high-level strategy guidance and (2) to execute and adapt the specified strategy [22].
A strategic action plan constrains and limits behaviors and actions of agents. It is obvious that a strategic action plan strongly influences performance of the team. IC, so, plays a major role in defining good strategy, smartly executing strategies, monitoring the situation, and refining and adjusting strategies to adapt crisis situation.

3.4.5.1 High-level strategy

High-level strategy guidance enables an IC as a human planner to express and encode his or her intuition for SAP. A strategy is composed of a set of parallel threads which are prioritized from high to low according to their importance. Furthermore, threads can operate in parallel during execution based on agent availability. A thread, itself, is composed of a unique ranking, a sub-team (a subset of agents), a sub-objectives (a subset of task types), and sub-locations (a subset of geographic objects). Agents may engage in several threads because of restricted resources.
A strategy decomposes a difficult and complex problem into simpler problems that can be solved by traditional AI techniques and automated systems. In another word, high-level strategy guidance partitions and decomposes the whole problem space into a set of small problems under human supervision. Decomposition of a coordination problem into some threads generates two new types of interdependency among threads: 1) agents who are shared among threads and 2) "enabling" dependencies that are formed among GTM tasks of threads.

3.4.5.2 Strategic action plan

A strategic action plan is the problem of appropriate assignments of agents to threads in a definite time. Because agents are shared among threads, an agent should be allocated to only one thread in a time. As a result, it is necessary for the IC to dynamically execute the specified strategy and adapt the made strategic plan regarding new disaster situation and availability of agents by optimally assigning agents to threads or smartly releasing agents from their thread into the next thread.
Assignment of a specific thread to a specific agent forces that agent to adapt his behaviors and actions regarding the thread definition. The made strategic plan will be sent from operation center (incident command post) to each agents so that they can make their

own tactical plan/decision for distributed multiagent coordination for doing tasks in the tactical level.

3.4.6 Action/Tasks Schedule

Strategic action scheduling estimates operation time (makespan) and assign (allocate) agents to time-consuming GTM tasks according to the made strategic action plan. Strategic action scheduling does not fully schedule detailed actions of agents for doing tasks in the tactical level.
The structure of a strategic action schedule contains assignment (allocation) information as follows: (1) location (geographic object) of the GTM task; (2) task type of the macro task; (3) start time; (4) finish time; (5) amount of tasks which are going to be done during this duration; (6) a set of agents who are assigned to this schedule; and (7) amount of dependent tasks which will be released/discovered within this location. The key point is that strategic action scheduling can be applied to different geographic layers.
Because of the geospatial, temporal, and macro aspects of tasks, strategic action scheduling takes into account eight rules. (1) More than one agent can be assigned to a GTM task, in fact, assigned agents form a coalition to execute this decision cooperatively; (2) Assignments are dynamic. It means that over time, new agents can join the coalition which have been scheduled for the GTM task; (3) Agents assigned to a GTM task are kept for that task until all task are accomplished; (4) Scheduling should follow the made action strategic plan; (5) Agents need a considering amount of travel time to reach a GTM task by moving from one point to another via the road network; (6) heterogeneous agents provides different capabilities which are required for heterogeneous tasks; (7) A coalition formed by many and professional agents can do a GTM task faster than another coalition; (8) GTM tasks may have a dynamical number of enabled tasks and a dynamical number of disenabled tasks because some agents may complete some tasks while other agents may release new tasks.

3.5 Data Modeling of the SAP Problem

3.5.1 Background to Data Modeling

The software engineering uses the data model in two related senses: 1) description of the objects represented by a computer system together with their properties and relationships and 2) a collection of concepts and rules used in defining data models. The main aim of data models is to support the development of information systems

by providing the definition and format of data. Data models are categorized to four types: 1) database model, 2) data structure diagram, 3) entity-relationship model, 4) geographic data model, 5) generic data model, and 6) semantic data model.

Data Structure Diagram is a diagram of the conceptual data model which documented the entities and their relationships, as well as the constraints that connects to them. That are several data modeling languages (and data modeling diagrams) that include the Unified Modeling Language (UML). The UML is a standardized general-purpose modeling language in the field of software engineering. It includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

3.5.2 The SAP Data Model

The SAP data model is the contribution of this chapter. Data modeling of the SAP problem, which was analyzed in the previous section, results in a SAP data model. **Fig. 3.5** and **Fig. 3.6** show the UML class diagram of this data model. Classes and relationships are presented in Fig. 3.5, and the attributes of these classes are shown in Fig. 3.6.

3.5.3 Describe the SAP Data Model

To have a better understand of the SAP data model, detailed information is given by the following subsections. It helps us to get better insight of classes, relationship among classes, and types of information provided by a certain class. It provides basic and fundamental knowledges for us to design and implement problem-solving algorithms. Future works such as [30**?**] need to know how to get and extract required data from the data model, how to design algorithms, how to deal with a database that support the SAP information, etc.

We classified the designed classes into the eight groups to model, formulate, and present the eight dimensions of the SAP problem as **Table 3.2** shows.

3.5.3.1 Urban search and rescue domain

Modeling the problem domain is done using the “taskType”, “capabilityType”, and “taskDependency” classes. Instances of these classes are initially defined by human users.
The “capabilityType” class is used to define a set of capability types which the problem

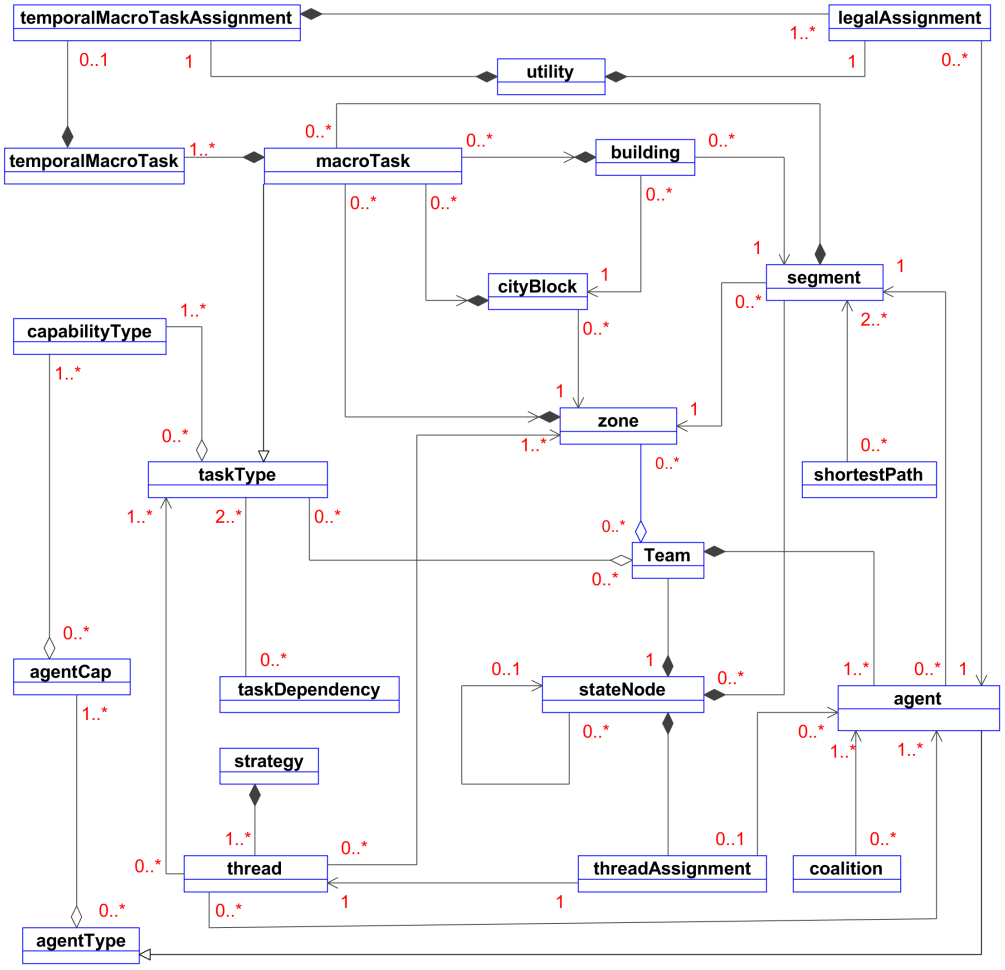


FIGURE 3.5: The SAP data model based on an UML class diagram: classes and relationships.

.

domain requires. Capabilities are applied for two purposes: 1) capabilities are required by tasks and 2) each agent provides a subset of capabilities. The "capabilityType" class connects the "taskType" class with the "agentCap" class. It, so, enables the system to identify or recognize what agents are eligible to take part in carrying out what tasks.

The "taskType" class defines all types of task which the problem domain includes. A task type needs a collection of capabilities (List <string> CapTypeIds) to do it. A specific task type may need either only one definite capability or several synchronous (simultaneous) capabilities in order to accomplish one unit of this task-type in a considering duration of time (the "duration" attribute). Duration of a task type includes uncertain information with the "p" probability.

The "taskDependency" class states dependency relationships among task types. It abstractly formulates all interdependencies among GTM tasks that are contained within a geo-object in common. The "enabling" dependency and the "facilitating" dependency between tasks are considered in coordination of agents.

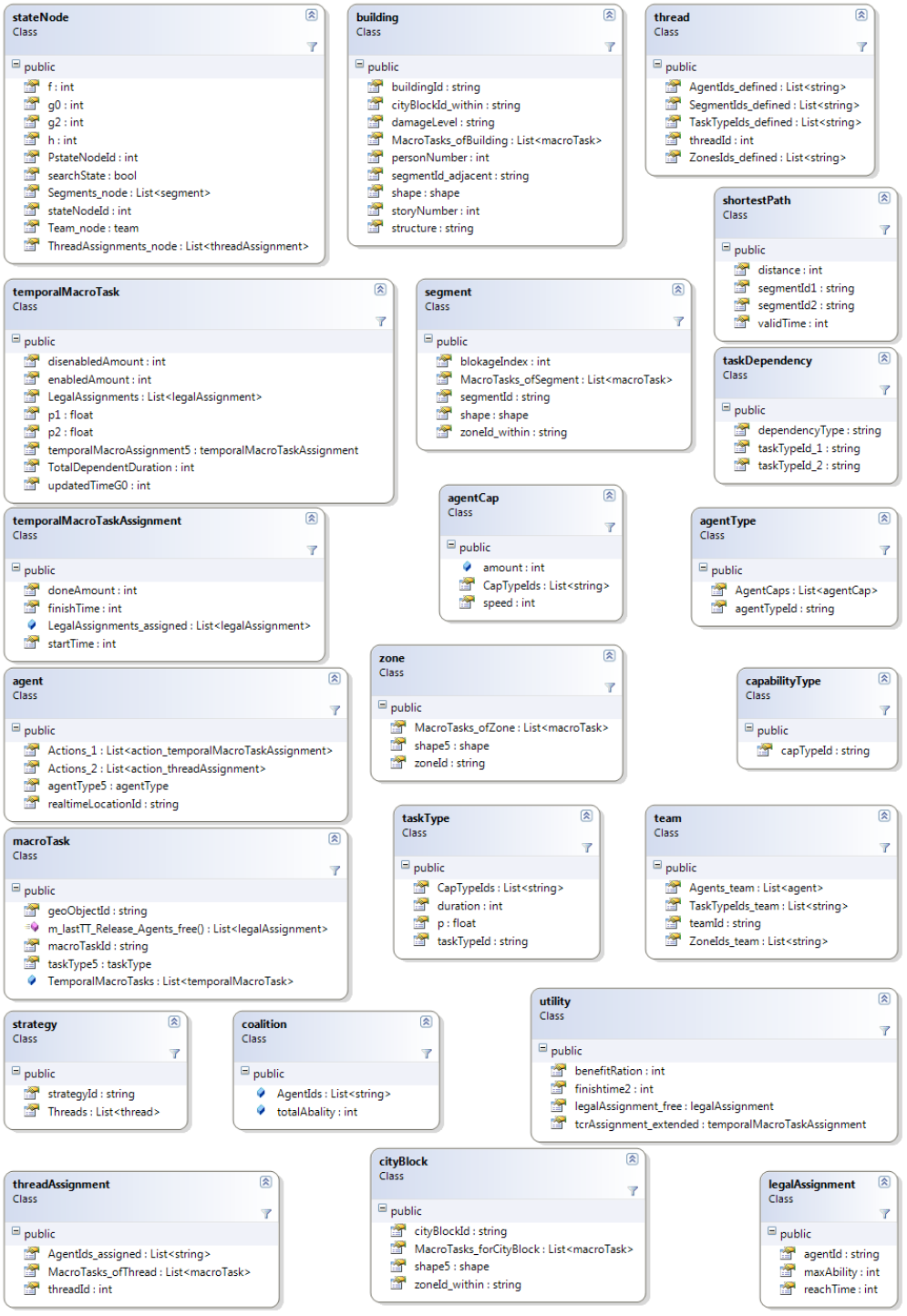

stateNode
Class
public
f : int
g0 : int
g2 : int
h : int
PstateNodeId : int
searchState : bool
Segments_node : List
stateNodeId : int
Team_node : team
ThreadAssignments_node : List<threadAssignment>
building
Class
public
buildingId : string
cityBlockId_within : string
damageLevel : string
MacroTasks_ofBuilding : List<macroTask>
personNumber : int
segmentId_adjacent : string
shape : shape
storyNumber : int
structure : string
thread
Class
public
AgentIds_defined : List<string>
SegmentIds_defined : List<string>
TaskTypeIds_defined : List<string>
threadId : int
ZonesIds_defined : List<string>
shortestPath
Class
public
distance : int
segmentId1 : string
segmentId2 : string
validTime : int
temporalMacroTask
Class
public
disenabledAmount : int
enabledAmount : int
LegalAssignments : List<legalAssignment>
p1 : float
p2 : float
temporalMacroAssignment5 : temporalMacroTaskAssignment
TotalDependentDuration : int
updatedTimeG0 : int
segment
Class
public
blokageIndex : int
MacroTasks_ofSegment : List<macroTask>
segmentId : string
shape : shape
zoneId_within : string
taskDependency
Class
public
dependencyType : string
taskTypeId_1 : string
taskTypeId_2 : string
agentCap
Class
public
amount : int
CapTypeIds : List<string>
speed : int
agentType
Class
public
AgentCaps : List<agentCap>
agentTypeId : string
temporalMacroTaskAssignment
Class
public
doneAmount : int
finishTime : int
LegalAssignments_assigned : List<legalAssignment>
startTime : int
zone
Class
public
MacroTasks_ofZone : List<macroTask>
shape5 : shape
zoneId : string
capabilityType
Class
public
capTypeId : string
agent
Class
public
Actions_1 : List<action_temporalMacroTaskAssignment>
Actions_2 : List<action_threadAssignment>
agentType5 : agentType
realtimeLocationId : string
taskType
Class
public
CapTypeIds : List<string>
duration : int
p : float
taskTypeId : string
team
Class
public
Agents_team : List<agent>
TaskTypeIds_team : List<string>
teamId : string
ZoneIds_team : List<string>
macroTask
Class
public
geoObjectId : string
m_lastTT_Release_Agents_free() : List<legalAssignment>
macroTaskId : string
taskType5 : taskType
TemporalMacroTasks : List<temporalMacroTask>
utility
Class
public
benefitRation : int
finishtime2 : int
legalAssignment_free : legalAssignment
tcrAssignment_extended : temporalMacroTaskAssignment
strategy
Class
public
strategyId : string
Threads : List<thread>
coalition
Class
public
AgentIds : List<string>
totalAbality : int
cityBlock
Class
public
cityBlockId : string
MacroTasks_forCityBlock : List<macroTask>
shape5 : shape
zoneId_within : string
threadAssignment
Class
public
AgentIds_assigned : List<string>
MacroTasks_ofThread : List<macroTask>
threadId : int
legalAssignment
Class
public
agentId : string
maxAbility : int
reachTime : int

FIGURE 3.6: The SAP data model based on an UML class diagram: Attributes of classes.

3.5.3.2 Team

The aim of this dimension is to model the structure of a disaster response team using four classes: “agentCap”, “agentType”, “agent”, and “team”. The IC of a team initially specifies the structure of his team.

TABLE 3.2: Grouping the classes of the SAP data model with regard to the components that constitute the SAP problem.

Dimensions of the SAP Problem	**Classes of the SAP Data Model**
1- Problem Domain	capabilityType taskType taskDependency
2- Disaster Response Team	agentCap agentCap agent team
3- Geospatial Information	building cityBlock segment zone shortestPath
4- Geospatial-Temporal Macro Tasks	macroTask temporalMacroTask
5- Human High-level Strategy Guidance	thread strategy
6- Strategic Action Plan	threadAssignment
7- Strategic Action Schedule	temporalMacroTaskAssignment legalAssignment
8- States of the World	stateNode

The "agentCap" class formulates all kinds of capabilities (skills or actions) which agents may possess within the team in the problem domain. An instance of the "agentCap" class indicates that how many (the "amount" attribute) of what capability or capabilities (List <string> CapTypeIds) are done with what speed.
The "agentType" class classifies agents according to their capabilities into different agent types. An agent type may have one capability type or several ones (List <agentCap> AgentCaps). It, also, indicates that in a time, an agent can select and use one capability type of its all capability types as its action.
The goal of the "agent" class is to present all agents which the team consists of. An agent has a certain agent type that specifies its capabilities domain. Because of uncertainty in data, approximate and real time location ("realtimeLocationId") of agents is defined by an adjacent segment (street segment).
The down level of the team is composed of a number of agents (List <agent> Agents_team). Because a team includes a subset of goals in a part of the disaster-affected area, the "team" class is associated with the "zone" and "taskType" classes.

3.5.3.3 Geospatial information

The "building", "cityBlock", "segment", "zone", and "shortestPath" classes organize some important geographic layers. The major role of GIS, here, is to provide efficient

tools for the incident command post to implement, develop, share, and manage a geospatial database that contains geographic information and location-based data.
Instances of these classes are geospatial objects which are distributed in the area. These classes enable six goals: 1) geo-visualize geographic information on GIS thematic maps, 2) present and manage non-spatial information which is associated to geographic information, 3) present spatial relationships, 4) compose a set of macro tasks for each geographic object, 5) integrate tasks information from one layer for another, and 6) view and percept tasks information in different geographic scales. Each geographic object comprises a set of macro tasks (List <macroTask>).
Each instances of the "shortestPath" class indicates a shortest distance that is between centers of two segments in a road network in a definite time. GIS calculates these instances for the road network. Blockage states of roads change over time because some blocked roads are cleared by road-clearing teams (bulldozers).

3.5.3.4 Geospatial-Temporal macro tasks

The "macroTask" and "temporalMacroTask" classes model and formulate geospatial-temporal macro (GTM) tasks.The Directed Acyclic Graph (DAG) was applied to present GTM tasks. A set of tasks with dependencies can be modeled by a DAG that is a directed graph with no directed cycles [19]. A task represents a vertex in the DAG; a directed edge (u,v) represents the dependency between two tasks and implies that task u must be completed before task v. Other complement information such as costs, duration, and deadline can be defined for each vertex.
A macro task has a definite task type. An macro task is composed of a set of temporal macro tasks (List <temporalMacroTask> TemporalMacroTasks) that specify a series of quantitative information of these accumulated tasks which are located within a definite geographic object (the "geoObjectId" attribute).
An instance of the "temporalMacroTask" class specifies quantitative information about its macro task. It defines how many enabled tasks (the "enabledAmount" attribute) and how many disenabled tasks (the "disenabledAmount" attribute) with a specific task-type are observed or estimated in a specific time (the "updatedTimeG0" attribute). Any change in quantitative information ("enabledAmount" or "disenabledAmount") leads to a new instance of "temporalMacroTask". An instance of this class may contain one instance of the "temporalMacroTaskAssignment" which is calculated by strategic action scheduling algorithms.

3.5.3.5 Human high-level strategy

The "strategy" and the "thread" classes express and encode human high-level strategy guidance. These classes enable the IC to specify a high-level strategy which is used in strategic action planning.
A strategy is composed of a collection of threads (List <thread> Threads) that encode and formulate human's intuition. A strategy partitions a complex planning problem into some interdependent, parallel, and prioritized threads.
A thread defines a sub-problem. It essentially comprises four attributes: 1) the "threadId" attribute: a unique ranking id that shows its priority among other threads, 2) the "AgentIds_defined"attribute: a subset of agents who are permissible to act in the thread domain, 3) the "TaskTypeIds_defined" attribute: a subset of task types that define a goals domain, and 4) the "ZoneIds_defined" attribute: a subset of zones that define a geographic scope. Associations of the "thread" class with the "agent", "taskType", and "zone" classes are necessary relationships to model this dimension.

3.5.3.6 Strategic action plan

The "threadAssignment" class formulates a strategic action plan. Assignment (allocation) of agents to threads results in a strategic action plan. This process is done either by the related algorithm or by human planners.
A strategic action plan is enclosed by the "stateNode" class that it enables the system to extract information of start time or finish time of the made strategic action plan.
For each thread, there is an instance of the "threadAssignment" class in a definite time. This class is associated with a subset of permissible agents (List <string> AgentIds_assigned) who are assigned to this thread.
It is important to take into account three facts. 1) An agent can be associated with one thread in a time. 2) Thread assignment is not a scheduling problem. 3) Agents, who are assigned to a particular thread, are responsible for doing a subset of tasks which their thread defines.

3.5.3.7 Strategic action schedule

The "temporalMacroTaskAssignment" and "legalAssignment" classes model a strategic action schedule. Instances of these classes are calculated by strategic action scheduling algorithms.

An entity of the class "temporalMacroTask" class can comprise an entity of the "temporalMacroTaskAssignment" class. This class consists of four properties: 1) the "startTime" attribute: start time of operation, 2) the "finishTime" attribute: finish time of operation, 3) the "doneAmount" attribute: a number of tasks which are estimated to be done, and 4) the "LegalAssignments_assigned" attribute: a subset of legal assignments. The "legalAssignment" class is associated with the "temporalMacroTaskAssignment" class and the "agent" class. A legal assignment states an assignment of an agent to a GTM task. This class class includes other detailed information such as maximum capability which an assigned agent can provide for the GTM or time which an assigned agent can start doing the GTM tasks.

3.5.3.8 State of the world

The "stateNode" class is used to model and simulate states of the environment. This class is used by search algorithms to find an optimal strategic action plan.
The class includes dynamic elements of the SAP problem. A state node essentially includes five properties: 1) start time, 2) finish time, 3) a team, 4) a strategic action plan that is made and valid during its duration, 5) the segments with their own macro tasks for strategic action scheduling, and 6) the parent node id.

3.6 Evaluation of the SAP Data Model

3.6.1 Objective of Evaluation

It is necessary to evaluate the quality of the SAP problem data model especially from the completeness point of view. The completeness factor refers to whether the data model contains all information required to support the required functionality of the system [26]. We consider some questions including 1) whether this data model is applicable?, 2) how to use the SAP?, 3) how data are produced by who (human, algorithms, or other information systems) and when?, 4) how to present/visualize the data, 5) whether this data model successful satisfies the defined requirement, 6) how a system or problem-solving algorithms can be developed using this model, and etc?

This section aims to show that the SAP data model is practical, useful, and efficient in development of SAP problem-solving approaches. The evaluation consists of several steps that include proposing, developing, and applying the GICoordinator [30] for a simulated scenario. This section dedicated to description of these steps.

3.6.2 GICoordinator: A GeoSpatial Intelligent Coordinator

GICoordinator, as a GIS-based intelligent assistant system, assists an incident commander in coordination of a team of field units in the domain of disaster emergency response in especial in urban search & rescue operations [30]. This spatial intelligent system supports the IC with intelligent algorithms for action planning and task scheduling for centralized coordination of a team in a dynamic and spatial environment [31**?**]. Also, it applies a spatial database for geographic and location-based information management and uses GIS functions to support development of these spatial intelligent algorithms. An IC is equipped with a computer that runs an instance of GICoordinator. An community (society or organization) of distributed GICoordinators presents distributed teams [32]. A simple version of this agent has been developed for the field units [**?**].

Development of the proposed approach includes development of the GICoordinator, development of a spatial database, implementation of a base GIS, and integration GIS with the GICoordinator. The SAP problem data model was applied by an object-oriented programming language to develop the GICoordinator. The SAP problem data model was implemented by the Microsoft SQL server to develop a base spatial database for achieving four objectives: 1) organize data (geographic information and non-geographic information) , 2) implement a base GIS, 3) integrate GIS with the GICoordinator, and 4) support human-GICoordinator interaction. A base GIS was connected to the database to provide efficient GIS tools for the IC. Integration of the GIS with the GICoordinator is done by this database which is shared between them.

3.6.3 Simulation of a SAP Problem

Simulation, as a method, enables us to apply the proposed approach for a SAP problem data model simulated for an USAR scenario. Simulation includes the following steps: 1) prepare geospatial information, 2) define the domain problem, 3) initiate a team, and 4) estimate GTM tasks using some loss estimation models [24],[25]. Related data are defined or initialized by the human planner and are inputted to the spatial database.
A part of the district 17 of the city of Tehran was chosen to be the case study whose area equals to 0.62 square kilometers. Geographic layers of this area, which are prepared in GIS, were exported to the spatial database. Shortest distances among centers of road segments and topology relationships among geographic information were extracted using abilities of GIS.
Next step was to define information of the USAR domain, which is composed of five task types. **Table 3.3** presents the task types and their capabilities requirements.

TABLE 3.3: Task types of the problem domain.

		Capability Requirements					
Task-Types	**Δ t**	***C0***	***C1***	***C2***	***C3***	***C4***	***C5***
T0	10	1	0	0	0	0	0
T1	20	0	1	0	0	0	0
T2	30	0	0	1	0	0	0
T3	70	0	0	0	1	0	0
T4	110	0	0	0	0	1	0
T5	1	0	0	0	0	0	1

Capabilities Description:
C0: C0-Reconnaissance
C1: C1-Search
C2: C2-SlightRescue
C3: C3-MediumRescue
C4: C4-HeavyRescue
C5: C5-TransportByVehicle

Task Types Description:
T0: T0-Reconnaissance
T1: T1-Search
T2: T2-SlightRescue
T3: T3-MediumRescue
T4: T4-HeavyRescue
T5: T5-MedicalTransportation

TABLE 3.4: Agent types and their features

		Capabilities					
Agent-Types	**speed**	***C0***	***C1***	***C2***	***C3***	***C4***	***C5***
A0-Reconnaissance	4	1	0	0	0	0	0
	1	0	0	0	0	0	1
A1-CanineSearch	1	1	0	0	0	0	0
	3	0	2	0	0	0	0
A2-ElectronicSearch	1	1	0	0	0	0	0
	2	0	1	0	0	0	0
A3-SlightRescue	1	1	0	0	0	0	0
	1	0	1	0	0	0	0
	4	0	0	1	0	0	0
A4-MediumRescue	1	0	0	1	0	0	0
	2	0	0	0	1	0	0
	1	0	1	0	1	0	0
A5-HeavyRescue	1	0	0	0	1	0	0
	3	0	0	0	0	1	0
A6-Ambulance	1	0	0	0	0	0	4
A7-Volunteer	1	0	0	0	0	0	1
	1	0	0	1	0	0	0

A flexible team was initiated for USAR. It consists of 12 agents. **Table 3.4** shows the agents and capabilities which they provide. It was assumed that all agents are free and are located in the incident command post in this snapshot. Location of agents is displayed in **Fig. 3.7**.

Final step was to estimate (simulate) macro tasks which are distributed in the area and are summarized and aggregated based on street segments. It was done in two steps: 1) estimate GTM tasks geo-located to buildings and 2) extract GTM tasks information

FIGURE 3.7: A thematic map of distribution of "search" macro tasks of the segments.

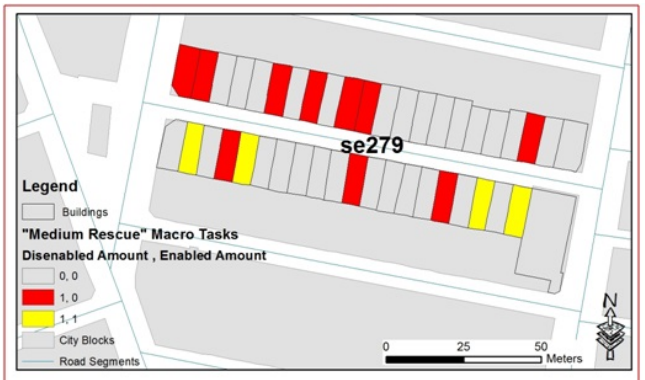
for segments using GTM tasks of buildings . First, GICoordinators executed a simple loss estimation algorithm to generate GTM tasks information for buildings. Then, an integration information algorithm was executed by GICoordinator to extract GTM tasks information of segments by summarizing and integrating GTM tasks information of buildings. Finally, the simulated operational area includes 1144 damaged buildings containing 5401 macro tasks, 53 roads segment including 297 macro tasks, and 4 operational zones with only 24 macro tasks. **Fig. 3.7** shows a map of the simulated SAP problem.

For an example, **Fig. 3.8** shows a map which highlights a segment and several buildings that are adjacent to the segments. 210 macro tasks estimated/observed for 43 buildings are integrated to extract at most 6 macro tasks of USAR operation for the segment 's316' at the time 0. The "T3-MediumRescue" macro task means that 4 medium rescue tasks (for rescuing 4 persons who have been located by search teams) are released and are ready to be done by appropriate teams along this segment, but 14 those are not enabled but are estimated to be released/discovered by "Search" operation. It is estimated that all of the 74 search tasks (17 enabled amount plus 57 disenabled amount) have to be accomplished in order to release (get enabled) all of the 35 disenabled rescue tasks.

Fig. 3.9 presents an algorithm to extract GTM tasks information for a definite segment by integrating GTM tasks information of building which are near this segment. In real world, city blacks have spatial topological relationships with zones objects.

3.6.4 Visualize the Simulated SAP Problem

Abilities and analytical tools of the GIS enable the IC to visualize and analyze the SAP problem information in order to get better awareness/perception of the global situation. In additional, the GICoordinator provides some data integration and information fusion



"Medium Rescue" Macro Tasks of the segment "se279"

taskTypeId	disenabledAmount	enabledAmount	updatedTimeG0
T0-Reconnaissance	0	43	0
T1-Search	57	17	0
T2-SlightRescue	8	1	0
T3-MediumRescue	14	4	0
T4-HeavyRescue	13	26	0
T5-MedicalTransportation	47	0	0

FIGURE 3.8: A GIS map of the macro tasks of the segment "se279".

algorithms/tools applied to the database.

Fig. 3.7 and **Fig. 3.10** present GIS-created thematic maps of the simulated SAP problem. It consists of the zones layer, the segments layers, location of 12 heterogeneous agents, and spatial distribution of macro tasks. This map models and presents the environment/situation which the IC is faced for coordination of actions of agents in order to accomplish tasks in a minimum time.

3.6.5 Apply the System for Strategic Action Planning and Scheduling

There are two significant benefits to apply the GICoordinator for strategic action planning problems: 1) human-machine collaboration for making a new strategic action plan & schedule and 2) automated update of the made strategic action plan. GICoordinator tries to provide a solution for the work flow presented in **Fig. 3.2**.

3.6.5.1 High-level strategy guidance specification

SAP, which results in a new strategic action plan and a new strategic action schedule, is done by collaboration between the GICoordinator and the IC. First, the IC specifies a strategy after he or she perceives the problem space using the GIS. Strategy information is inputted into the GICoordinator via the provided interface.

Table 3.5 shows the human's strategy for coordination of the team. The strategy is composed of four threads that formulate and encode the human's intuition and initiative

```
Data: B :a set of buildings.
Data: Id :the Id of a segment.
Data: TT :a set of task types.
Result: MT5 : a set of macro-tasks extracted for the segment.
begin
    MT1 ⟵ ∅
    MT2 ⟵ ∅
    for b ∈ B do
        if b.segmentId_adjacent = Id then
            for macroTask ∈ b.MacroTasks do
                MT1 ⟵ MT1 ∪ {macroTask}
            end
        end
    end
    Group ⟵ Classify(MT1, TT)
    for G ∈ Group do
        Se ⟵ 0
        Sd ⟵ 0
        for macroTask ∈ G do
            Se ⟵
            Se+macroTask.TemporalMacroTasks.last().enabledAmount
            Sd ⟵ Sd +
            macroTask.TemporalMacroTasks.last().disenabledAmount
        end
        tmt0 ⟵ Create_anEmpty_temporalMacroTask()
        tmt0.enabledAmount ⟵ Se
        tmt0.disenabledAmount ⟵ Sd
        mt0 ⟵ Create_anEmpty_macroTask()
        mt0.taskType ⟵ G.taskType
        mt0.geoObjectId ⟵ Id
        mt0.TemporalMacroTasks ⟵
        mt0.TemporalMacroTasks ∪ {tmt0}
        MT2 ⟵ MT2 ∪ {mt0}
    end
end
```

FIGURE 3.9: A algorithm for integrating macro tasks information of buildings in order to extract new macro tasks information for a road segment.

for coordination of agents. The first thread states that the top-ranking objective for the team is to do reconnaissance operation located in two operational zones zo1, zo2. To accomplish this objective and do relevant tasks, any subset of the six agents ag0A0, ag1A0, ag2A1, ag3A1, ag4A2, ag5A2 is allowed to be assigned to this objective. The second thread expresses the second-ranking objective that is to carry out all search operation contained within the same operational zones; and any coalition composed of the six agents ag2A1, ag3A1, ag4A2, ag5A2, ag6A3, ag7A3 are permitted to be allocated to this thread. The third thread specifies that three kinds of rescue operation should be performed within the zone 1 and the zone 2. To achieve this objective, any subset of six agents can be assigned to this thread-definition. The last thread presents the

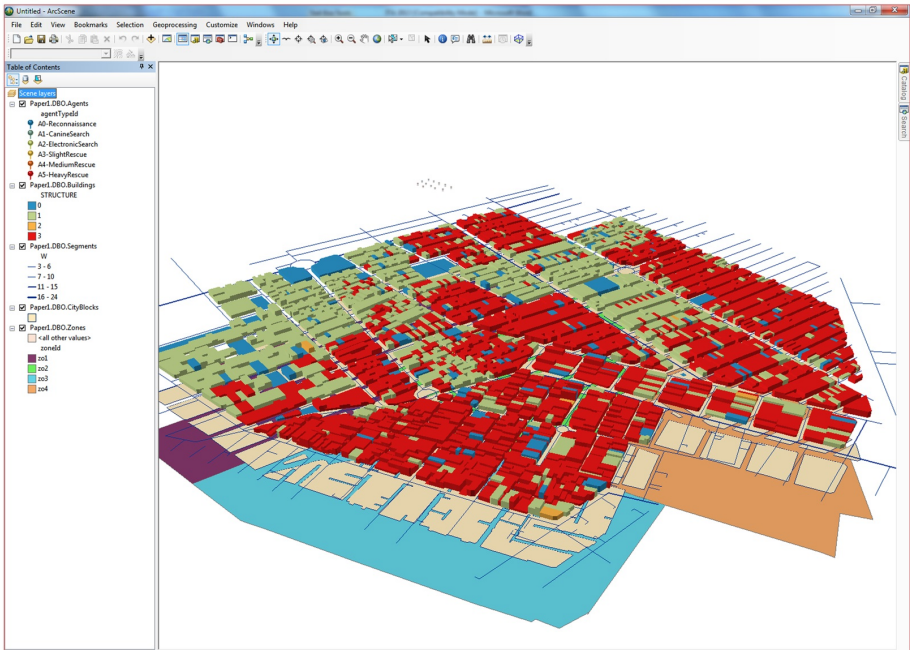


FIGURE 3.10: Using GIS for 3D visualization of agents, buildings, street networks (segments), city blocks, and zones geographic layers that compose the region 17 of Tehran.

low-ranking goal which the IC defined for the team. It states that all five task-types of USAR which are within two zones 3 and 4 can be done by any subsets of six agents. The four thread-definitions partition the SAP problem into four prioritized, parallel, and dependent sub-problems.

To specify the four threads, the IC takes into account two important facts: 1) agents availability and 2) enabled tasks. The agents shared among threads and the tasks dependencies among thread-definitions make the four threads interdependent completely or partially. For an example, agent "ag4A2", who has the different capabilities, is defined and shared among the first, second, and fourth thread-definitions. Rules let this agent, first, be assigned to the first thread-definition, and then this thread releases this agent into the second thread as the next, specific, and lower-ranking thread-definition. Consequently, it takes time for the agent "ag4A2" to become available for the fourth thread in order to accomplish its objective. For another example, the second thread is completely dependent to the first one. Reconnaissance operations specified by the first one release/enable the search operation defined by the second one. Agents assigned to the third thread have to wait for agents who can release/discover their rescue tasks by accomplishing search operation defined by the the second thread-definition.
There is an interesting point that the actions domain of a definite agent such as "ag10A5 is limited and identified by thread-definitions in time, space, and goal. For example, the third thread lets this agent do only three types of rescue operations within only two

TABLE 3.5: A high-level strategy composed of 4 threads which is specified by the incident commander for strategic action planning and scheduling.

thread Id	**{zone}**	**{task-type}**	**{agent}**
1	zo1 zo2	T0-Reconnaissance	ag0A0 ag1A1 ag2A1 ag3A1 ag4A2 ag5A2
2	zo1 zo2	T1-Search	ag2A1 ag3A1 ag4A2 ag5A2 ag6A3 ag7A3
3	zo1 zo2	T2-SlightRescue T3-MediumRescue T4-HeavyRescue	ag6A3 ag7A3 ag8A4 ag9A4 ag10A5 ag11A5
3	zo3 zo4	T0-Reconnaissance T1-Search T2-SlightRescue T3-MediumRescue T4-HeavyRescue	ag0A0 ag2A1 ag4A2 ag6A3 ag8A4 ag10A5

zones 1 and 2. If this autonomous agent is assigned to third thread-definition, he will have a partial autonomy.

3.6.5.2 Optimal assignments of agents to threads

The second step for SAP and making a strategic action plan is to execute the human-specified strategy by optimally assigning the 12 free (released) agents to the 4 threads according to the SAP problem information and the human' strategy. Quality of thread-assignments affects the operation time (makespan) for the whole problem. Therefore, the key question arises here: which idle agents should be assigned to which thread in this time? Is it a smart choice for a thread to keep a maximum number of available agents for itself and release unwanted agents into the next thread?
Because this decision-making is difficult for the human (IC), the GICoordinator runs a set of sophisticated search algorithms to find an optimal solution for this problem. Calculated information includes three types of information: 1) a number of temporal sets of strategic action plans (the "threadAssignment" class), 2) a number of temporal sets of strategic action schedules ("temporalMacroTaskAssignment" and "legalAssignment"), and 3) overall operation time. These data update the spatial database and are presented for the IC to evaluate and understand the evolution of the defined strategy.

TABLE 3.6: Two temporal strategic action plans (allocation of agents to threads) calculated by the assistant software agent.

thread Id	**Agent Allocation to Threads at Time**	
	0	***579***
1	ag0A0 ag1A0 ag2A1 ag3A1 ag4A2 ag5A2	ag3A1 ag4A2 ag5A2
2	ag7A3	ag7A3
3	ag9A4 ag11A5	ag9A4 ag11A5
4	ag6A3 ag8A4 ag10A5	ag6A3 ag8A4 ag10A5 ag0A0 ag2A1

This functionality of the GICoordinator supports the IC to make a good strategy by refining and adjusting his strategy.

Table 3.6 shows two temporal and optimal strategic plans by optimal assignments of agents to the threads. They are computed by the GICoordinator. The first strategic plan is made for the current time "0"; it states that the GICoordinator has decided to assign ag0A0, ag1A0, ag2A1, ag3A1, ag4A2, ag5A2 as a maximum and possible number of agents to the first thread, and then it released the remained agents into the second thread-definition. But, it calculated that the best decision is to keep only the agent "ag7A3" for this thread and send others to the next thread. All available agents are assigned to the last thread. The search algorithms estimate that the current strategic is valid until time 579, and the second new strategic plan made should start at this time. The GICoordinator adapted the strategic plan with the updated states of the world. The result is to release the agents "ag0A0", "ag1A0" from their thread.

Strategic action plans of the agents are extracted from the SAP problem data. E.g. the strategic action plan of the agent "ag2A1" states that this agent, first, is sent to the zone 1 and 2 to do reconnaissance operation from time 0 until 560, and then it has to attend the fourth thread to perform five types of USAR operation located in the the zone 3 ,4 from time 579 to time 596. This plan estimates that this agent will not be used by the team in USAR after this time anymore.

3.6.5.3 Strategic action scheduling

The GICoordinator calculates a new strategic action schedule by assignment of GMT tasks to agents. Whenever a new strategic action plan is made, the GICoordinator runs a set of heuristic algorithms to assign macro tasks of segments to agents considering the

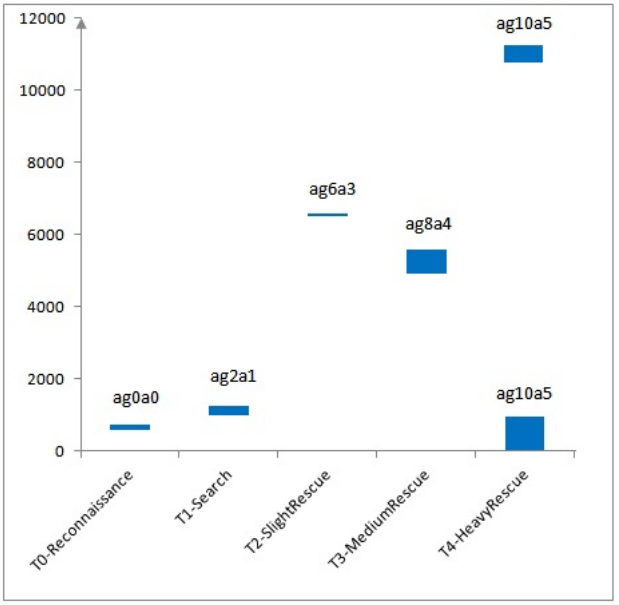


FIGURE 3.11: A graph of temporal assignment of agents to macro tasks of segment "se279" as a strategic action schedule.

simulated SAP data and the strategic plan. These algorithms change the state of the worlds (agents, tasks).

We selected the segment "se279" to present an example of strategic schedule information. **Fig. 3.11** shows temporal assignments of agents to macro tasks located in this segment.

Using the GIS, information of strategic action schedules of agents is visualized on thematic 3D map to make a good vision of the SAP problem.

3.6.6 Automated adaption of the strategic action plan

Uncertain and dynamic situations in USAR environment disturb the current strategic action plan, which is made for the current time and is executed by the agents. It is necessary to refine and adapt this strategic plan to new situations. Therefore, three key questions arise in this requirement: 1) is a right time to release agents from their thread and make a new strategic plan? 2) Which agent should leave its thread-definition? 3) Which thread is required to release its agents?

The role of the GICoordinator is to continuously and autonomously monitor the SAP problem data, which are updated over time, in order to address these questions. If the strategic plan is required to be adapted, the GICoordinator will re-make optimal thread-assignments according to the updated the SAP problem.

The applied this feature in its search algorithms to recognize the right time for releasing agents from their thread.

3.7 Discussion

The SAP problem data modeling follows two objectives: 1) formulate and model the SAP problem for coordination of a disaster response team and 2) provide a framework to support development of intelligent software systems for making strategic action plans & schedules.

Efficiency of proposed approaches is dependent on two main criteria: 1) quality of data and 2) efficiency of algorithms. The SAP problem data model enables and supports implementation and development of these main criteria.

The SAP problem data model can be considered an appropriate framework or approach to be used for modeling other related SAP problems. Although this chapter focused and analyzed a specific SAP problem, the designed SAP problem data model can be modified and refined to address new requirements such as fire fighting teams, flood evacuation operations, resource allocation problems, medical emergency transportation operation, road-clearing vehicles, coordination problem among distributed ICs of multiple teams, new types of information, tactical decision making, or even military operations commander.

GTM tasks include uncertainty, simplicity, and approximation information because tasks are abstracted from a low geographic layer into a higher one. But they are important and useful for an IC to reach two goals: 1) efficient visualization of situation awareness and 2) fast but approximate calculation/estimation.

The grammar of threads, which encodes human's intuitions for solving complex problems, is essentially composed of the four types of information in the designed SAP problem data model. It makes possible to formulate other insights of ICs e.g. specification of new constrains or specification of the type of actions scheduling.

The "threadAssignment" class models information of allocation of coalitions to sub-tasks in sub-locations for a duration. Because of spatial topology relationships between geographic objects, strategic action plans can be extracted for different geographic layers. Information of this class can be integrated with internal beliefs and behaviors of agents. The SAP problem data model enables an information system to extract and mine new information from the database and display them for ICs. In addition, it can be used in development of disaster management system [28] and incident commander decision support systems.

3.8 Results

Fig. 3.5 and **Fig. 3.6** show the contribution of this chapter. The SAP problem data model addresses the two research questions stated in section 1. It includes five new

findings that denote the contribution and originality of this chapter.
First finding is to analyze the SAP problem and design the SAP data model for modeling the SAP problem data. The SAP problem data model is important and critical for develop any approach for strategic action planning in a team for coordination of disaster emergency response management.
Second one is to model geospatial-temporal macro tasks by the "macroTask" and "temporalMacroTask" classes and integrate with other classes. They are used for five purposes: 1) present a set of tasks summarized and distributed in geographic objects, 2) extract and integrate tasks information from one geographic layer for another layer according to their spatial topological relationships, 3) organize and manage temporal changes in tasks and estimate their effects on dependent tasks, 4) create thematic maps and extract new information, and 5) use for strategic action scheduling.
Third finding is to encode and formulate human high-level strategy guidance by the "thread" and the "strategy" classes and integrate them with the data model. These classes enable human agents engage in the planning process and collaborate with an automated information system. These two classes integrate human's intuition and initiative in the data model and enable the automated system to apply them for making an optimal strategic action plan.
Next finding is the "threadAssignment" classes to model a high-level strategic action plan. This class constrains agents' behaviors by allocating them to threads.
The "temporalMacroTaskAssignment" and the "legalAssignment" classes are the fifth finding. These classes present a strategic action schedule and are integrated with other classes to form a complete data model of the SAP problem.

3.9 Conclusion

The SAP problem data model tried to model the strategic planning problem in coordination of an emergency response team during emergency response management. It is required to develop intelligent software systems that collaborate with the humans to address the SAP problem by good strategy specification, optimally strategic action planning, and automated adaption. This data model is a focus on a specific problem, but it can be refined to address other ICs' requirements.

Future work includes three directions. First one is to address the problem-solving algorithms that intend to make strategic action plans and schedules. The proposed approaches will use the SAP data model. Second work may be to improve the SAP problem with regard to new requirements or demands e.g. resource allocation problem, coordination of distributed teams, or integrating the SAP data model with field units decision

support systems, The last idea is to integrate the spatial database of the GICoordinator with information systems and develop proper algorithms for data data/information extraction and integration in order to get required information for this system from distributed data warehouses.

References

[1] Bigley, Gregory A., and Karlene H. Roberts. "The incident command system: High-reliability organizing for complex and volatile task environments." Academy of Management Journal 44, no. 6 (2001): 1281-1299.

[2] Boddy, M., B. Horling, J. Phelps, R. P. Goldman, R. Vincent, A. C. Long, B. Kohout, and R. Maheswaran. "C_TAEMS Language Specification, Version 2.02." DARPA, Arlington, VA (2006).

[3] Buck, Dick A., Joseph E. Trainor, and Benigno E. Aguirre. "A critical evaluation of the incident command system and NIMS." Journal of Homeland Security and Emergency Management 3, no. 3 (2006).

[4] Burstein, Mark H., and Drew V. McDermott. "Issues in the development of human-computer mixed-initiative planning." Advances in Psychology 113 (1996): 285-303.

[5] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu Upadhyaya. "Design principles of coordinated multi-incident emergency response systems." In Intelligence and Security Informatics, pp. 81-98. Springer Berlin Heidelberg, 2005.

[6] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[7] Decker, Keith S., and Victor R. Lesser. "Quantitative modeling of complex environments." International Journal of Intelligent Systems in Accounting, Finance, and Management 2, no. 4 (1993): 215-234.

[8] Egenhofer, Max J., and Robert D. Franzosa. "Point-set topological spatial relations." International Journal of Geographical Information System 5, no. 2 (1991): 161-174.

[9] FEMA, "Fema Incident Action Planning Guide.", http://www.uscg.mil/hq/cg5/cg534/nsarc/FEMA%20Incident%20Action%20Planning%20Guide%20(IAP).pdf

[10] Fiedrich, Frank, Fritz Gehbauer, and U. Rickers. "Optimized resource allocation for emergency response after earthquake disasters." Safety Science 35, no. 1 (2000): 41-57.

[11] Fuhrmann, Sven, Alan MacEachren, and Guoray Cai. "Geoinformation technologies to support collaborative emergency management." In Digital Government, pp. 395-420. Springer US, 2008.

[12] Hunger, J. David, and Thomas L. Wheelen. Essentials of strategic management. New Jersey: Prentice Hall, 2003.

[13] OCHA, INSARAG Guidelines and Methodology Manual, www.usar.nl/upload/docs/insarag_guidelines_july_2006.pdf

[14] Jain, Sanjay, and Charles McLean. "Simulation for emergency response: a framework for modeling and simulation for emergency response." In Proceedings of the 35th conference on Winter simulation: driving innovation, pp. 1068-1076. Winter Simulation Conference, 2003.

[15] Jennings, Nick, Sarvapali D. Ramchurn, Mair Allen-Williams, Raj Dash, Partha Dutta, Alex Rogers, and Ioannis Vetsikas. "The ALADDIN project: Agent technology to the rescue." In Proceedings of the First Intl. Workshop on Agent Technology for Disaster Management. 2006.

[16] Johnson, Russ. "GIS technology for disasters and emergency management." An ESRI White Paper (2000).

[17] Khalil, Khaled M., M. Abdel-Aziz, Taymour T. Nazmy, and Abdel-Badeeh M. Salem. "Multi-agent crisis response systems-design requirements and analysis of current systems." arXiv preprint arXiv:0903.2543 (2009).

[18] Kitano, Hiroaki, and Satoshi Tadokoro. "Robocup rescue: A grand challenge for multiagent and intelligent systems." AI Magazine 22, no. 1 (2001): 39.

[19] Kwok, Yu-Kwong, and Ishfaq Ahmad. "Benchmarking and comparison of the task graph scheduling algorithms." Journal of Parallel and Distributed Computing 59, no. 3 (1999): 381-422.

[20] Maheswaran, Rajiv T., Pedro Szekely, Marcel Becker, Stephen Fitzpatrick, Gergely Gati, Jing Jin, Robert Neches et al. "Predictability & criticality metrics

for coordination in complex environments." In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, pp. 647-654. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[21] Maheswaran, Rajiv T., Craig M. Rogers, Romeo Sanchez, and Pedro Szekely. "Human-agent collaborative optimization of real-time distributed dynamic multi-agent coordination." In Workshop 25: Optimisation in Multi-agent Systems, p. 49. 2010.

[22] Maheswaran, Rajiv T., Pedro Szekely, and Romeo Sanchez. "Automated adaptation of strategic guidance in multiagent coordination." In Agents in Principle, Agents in Practice, pp. 247-262. Springer Berlin Heidelberg, 2011.

[23] Malone, Thomas W., and Kevin Crowston. "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) 26, no. 1 (1994): 87-119.

[24] Mansouri, B., K. A. Hosseini, and R. Nourjou. "Seismic human loss estimation in Tehran using GIS." In 14th World Conference on Earthquake Engineering, Beijing. 2008.

[25] Mansouri, Babak, Mohsen Ghafory-Ashtiany, Kambod Amini-Hosseini, Reza Nourjou, and Mehdi Mousavi. "Building seismic loss model for Tehran." Earthquake Spectra 26, no. 1 (2010): 153-168.

[26] Moody, Daniel L. "Measuring the quality of data models: an empirical evaluation of the use of quality metrics in practice." In ECIS, pp. 1337-1352. 2003.

[27] Musliner, David J., Edmund H. Durfee, Jianhui Wu, Dmitri A. Dolgov, Robert P. Goldman, and Mark S. Boddy. "Coordinated Plan Management Using Multiagent MDPs." In AAAI Spring Symposium: Distributed Plan and Schedule Management, pp. 73-80. 2006.

[28] Nakabayashi, Itsuki. "Disaster Management System for Wide-Area Support." Journal of Disaster Research 1: 46-71.

[29] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[30] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." In Workshop on

Robots and Sensors integration in future rescue INformation system, ROSIN'13. In conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), 2013.

[31] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC), 2014 (in press)

[32] Nourjou, Reza, and Michinori Hatayama. " Simulation of an Organization of Spatial Intelligent Agents in the Visual C#.NET Framework.", International Journal of Computer Theory and Engineering, 2014. (under review)

[33] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, and Pedro Szekely. "Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination." Journal of Software, 2014. (under review)

[34] Nwana, Hyacinth S., Lyndon C. Lee, and Nicholas R. Jennings. "Coordination in software agent systems." British Telecom Technical Journal 14, no. 4 (1996): 79-88.

[35] Schurr, Nathan, Janusz Marecki, John P. Lewis, Milind Tambe, and Paul Scerri. "The defacto system: Training tool for incident commanders." In AAAI, pp. 1555-1562. 2005.

[36] Vafaeinezhad, Ali Reza, Ali Asghar Alesheikh, Majid Hamrah, Reza Nourjou, and Rouzbeh Shad. "Using GIS to Develop an Efficient Spatio-temporal Task Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications–ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009.

Chapter 4

Intelligent Algorithms for Assignment of Field Units to Human Strategy

4.1 Abstract

Problem: Multi-agent coordination is an important issue in the domain of disaster emergency response operations where a a team of agents (field units or robots) aims to achieve a joint objective. Therefore the responsibility of incident commander is to 1) specify a good strategy composed of a number of threads (a set of prioritized sub-problems), 2) appropriately assign/allocate agents to these threads as a strategic decision for constraining agents to specific threads, and 3) timely release proper agents from assigned threads in order to adapt a strategic decision to the new situation.

Objective: The purpose of this chapter is to present an intelligent algorithm that assists the human in multi-agent coordination by providing two key issues: 1) automatically calculate and present a set of feasible alternatives for selecting a choice as a strategic decision in a definite time and 2) autonomously and timely identify a subset of assigned agents that should be released from their threads in order to refine a strategic decision.

Method: This algorithm expands a decision tree from an state node in which a thread (or several threads) has received a set of new agents either from the IC or from the higher thread. Each thread, which is associated with one level of the decision tree with a number of nodes, calculates a set of efficient coalitions using all available agents and generates a new node per a coalition to show which agents are allocated to this thread and which ones are released into the lower thread. In real time, this algorithm continuously

observes and monitors the task environment to identify a subset of assigned agents that can not provide any efficient capabilities for their threads and should be released into next threads.

Results: To get better insight, this chapter applied this algorithm for the coordination problem of a team in a simulated search & rescue scenario in an earthquake disaster-affected area where the team's goal is to save trapped people who are distributed in five operational zones. The result was an infinite set of alternative scenarios per a human-defined strategy. The calculated alternatives were presented for the IC to select a choice according to his intuition or to delegate the system to search for the optimal one.

4.2 Introduction

Review of the past disasters (natural or human-made) shows that coordination of disaster emergency/crisis response operations is a significant and essential issue. A responder team such as INSARAG (International Search and Rescue Advisory Group) [7] that comprises an IC (Incident Commander) and several field/operational units, which are called *agents* in this chapter, is faced with the problem of carrying out spatially dispersed tasks under evolving execution circumstances in a manner that achieves a joint objective in a minimum time. Coordination is the act of managing interdependencies among activities performed to achieve a goal [11]. To maximize the global payoff, five types of coordination are considerable to be manged within the team: 1) tasks dependencies, 2) actions dependencies, 3) redundant actions, 4) information sharing, and 5) agents allocation [15]. Inefficient coordination results in idle agents, conflict between actions, or redundant activities, and consequently operations take a very long duration to be completed. Effective coordination is hard and challenging because of characteristics which this domain includes such as: uncertainty in outcome of actions of agents, incomplete information of tasks information, time pressure, limited resources, task flow, etc [4].

Coordinating a team of different agents is the crucial issue within a team that their aim is to achieve a joint objective in the domain of disaster emergency response operations. Centralized multi-agent coordination is the key role of an IC in a team. Three main responsibilities of the IC include making three types of important decisions: 1) select a global objective and specify a strategy composed of a number of threads which are a set of prioritized sub-problems, 2) make a strategic decision by allocation of agents to these threads for constraining actions of agents to assigned threads, 3) identify a proper time in which the made strategic decision should be adapted to new situations by identifying a subset of assigned agents that should be released from their threads and be sent to

next threads [13], [14]. Human decisions define macro behaviors of the team because they do not explicitly specify micro actions which should be done by agents. A strategic decision specifies a domain of actions of agents in order to maximize the global payoff.

Calculation of feasible alternatives for making a strategic decision is a essential process. These alternatives enables the IC to understand what sufficient options are available in a time. In addition, in a real-time situation, identifying a right time in which to revise a strategic decision is a hard and challenging problem for the IC. These two issues are key for efficient coordination. It is an important issue to caBut it is a difficult and challenging problem for human because of the complexity of crisis emergency response. Also a full automated system can not provide any proper solution for this problem and is not considered a reliable approach for the human.

As a result, it is important and necessary to develop an ideal approach based on mixed-initiative approaches in which an intelligent system assists the IC in multi-agent coordination and collaborates with him in decision making. This chapter focuses on two sub-problems and the proposed issues will address them: 1) how to assist the IC and collaborate with human in appropriate assignment of agents to threads? and 2) how to support human in the adaption of a thread assignment to the new crisis situation in a right time?

However, many works have been developed for multiagent coordination especially for planning, scheduling, tasks assignment, action coordination, decision making, or optimization; unfortunately they do not provide a proper solution for the problem addressed by this chapter. There are deficiencies which are categorized as follows:

- They aim to fully specify micro actions of agents. But, it is impossible for an I.C to exactly make all decisions and define detailed actions of agents who are located in an uncertain and dynamic environment. These agents, which are human or robots, select and do a best action based on perception they have from their local environment with detailed information.
- They are automated systems that do not involve human in the loop. The main obstacle is scale as it is currently infeasible for a fully automated system to effectively reason about all the possible futures that may arise during execution of tasks in a complex environment.

The purpose of this chapter is to present an autonomous algorithm. Two main issues are of importance to apply this algorithm in the centralized multi-agent coordination by the IC: 1) automated calculation of a set of feasible alternatives for selecting a choice as the strategic decision in a definite time and 2) autonomous adaption of a strategic

decision in a right time by identifying a right subset of agents that should be released from a right subset of threads.

This chapter is organized as follows. To get better insight, section II provides our motivation in this chapter by introducing a simulated urban search & rescue scenario in which the IC of a team of four agents is faced with the coordination problem of agents. Section III reviews some related works. The problem is stated in Section IV in more detail. The approach and the proposed algorithm are presented in Section V. Section VI is dedicated to achieved results. Finally, we have a conclusion.

4.3 Motivation

Imagine that an earthquake disaster has occurred in an urban area which is displayed in Fig. 4.1. Urban search and rescue (USAR) as a major function of disaster emergency response operations aims to rescue the greatest number of people who are trapped under the debris of damaged buildings in the shortest amount of time. To save a victim in a certain spatial location, a sequence of three dependent location-based tasks should be accomplished simply, and each type of task requires a set of definite capabilities and a considerable amount of time. Table 4.I lists these task types and their capabilities requirements. We ignore the deadline for tasks in this chapter.

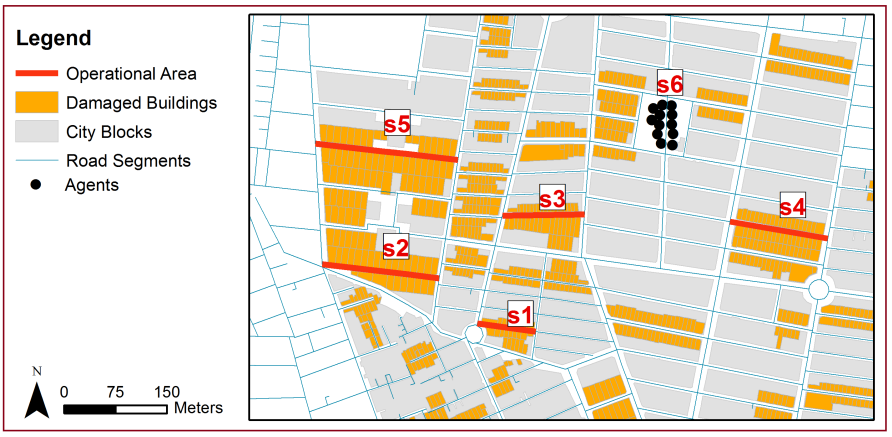


FIGURE 4.1: A GIS map of spatial distribution of location-based temporal macro tasks in five operational zones (road segments) of the disaster-affected area and location of the disaster response team in the initial state.

A team of four agents has been assigned to five operational areas which are presented by five road segments displayed in Fig. 4.1. Actions which an agent can do are presented by capabilities that this agent possesses. An agent can have different capabilities which are required by tasks. In a time, this agent can execute an action with a definite speed. Table 4.II lists characteristics of these agents. It was assumed the agents are *free* (or idle) and are located in the incident command post.

TABLE 4.1: Three Task Types of the Problem Domain.

Task-Type	**Δ *t* (minute)**	**Capability Requirements**		
		C0	***C1***	***C2***
T0	5	1	0	0
T1	20	0	1	0
T2	60	0	0	1

Capabilities Description:
C0: Reconnaissance
C1: Search
C2: Light Rescue

Task Types Description:
T0: Reconnaissance
T1: Search
T2: Light Rescue

TABLE 4.2: Capabilities/Abilities of Four Agents.

Field Unit ID	**action speed**	**Number of Capabilities**		
		C0	***C1***	***C2***
a0	2	1	0	0
a2	1	1	0	0
	2	0	1	0
a6	1	0	1	0
	2	0	0	1
a7	1	0	1	0
	2	0	0	1

Shortest distances among six road segments are calculated by GIS and are presented in Table 4.III. To do spatially distributed tasks, agents need to move from one location to another through the road network with the moving speed equal to 20. To make the problem simple, these variables do not change over time. These data are used by the IC in decision making. This table and related information are provided and are updated by relevant teams or organization whose activities are to clear road blockages.

TABLE 4.3: The Shortest Paths (given in meter) among Road Segments in the Geographic Area Visualized in Fig. 4.1.

	s2	**s1**	**s3**	**s4**	**s5**	**s6**
s2	0	225	447	764	364	625
s1	225	0	370	687	418	548
s3	447	370	0	343	452	221
s4	764	687	343	0	618	224
s5	364	418	452	618	0	476
s6	625	548	221	224	476	0

A set of LoTeM tasks (Location-based Temporal Macro tasks) are associated to each road segment. Simply, a LoTeM task is an aggregative task of all tasks with the same task type which are spatially located within a geographic area such as road segment, city block, etc [13]. Table 4.IV shows that twelve LoTeM tasks are located to five segments at the time 0. Information of these tasks forms the big picture that the IC observes from the task environment. For example, the 10th LoTeM task states that the proximity of

the the road segment "s5", five light rescue tasks are estimated to be revealed and six tasks are available and ready to be done. Also, it is estimated that if the 9th LoTeM task (2 not yet enabled amount plus 5 enabled amount) is completely accomplished, five light rescue tasks will be revealed in the same road segment. Finally, all these information is presented for the IC in order to provide a timely situational awareness of the world's state.

TABLE 4.4: A Set of Location-based Temporal Macro Tasks which Are Associated with Five Road Segments Displayed in Fig. 4.1.

No.	**Location (Road S.)**	**Task-Type**	**Not Yet Enabled Amount**	**Enabled Amount**
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	15
6	s3	T1	8	0
7	s3	T2	2	8
8	s5	T0	0	10
9	s5	T1	2	5
10	s5	T2	5	6
11	s4	T1	0	18
12	s4	T2	10	2

First step in coordination of agents by the IC is to specify a strategy. The joint objective that the IC selects for the team is to rescue all victims distributed in the five operational areas. Table 4.V presents the human strategy composed of three threads in order to achieve the defined goal. Thread 1 states that first priority of the team is to do reconnaissance and search operations at three geographic areas {s3, s4, s5}, and any *appropriate* subset (sufficient coalition) of four agents {a0, a2, a6, a7} can be assigned to this thread in order to accomplish tasks which are spatially distributed within these areas. The human strategy, also, has defined the agent a2 for three threads. To specify a strategy, the IC takes into account two important facts: 1) agents availability and 2) enabled tasks. Agents shared among threads and tasks dependencies among threads make threads interdependent completely or partially. For an example, the search LoTeM task of the first thread enables/reveals the light rescue LoTeM task of the second thread.

TABLE 4.5: An Example of Human Strategy with Three Threads Specified by the IC for Centralized Multi-agent Coordination.

Thread Id	**a sub-Location**	**a sub-Goal (Task Type)**	**a sub-Team**
1	s3, s4, s5	T0, T1	a0, a2, a6, a7
2	s3, s4, s5	T2	a2, a6, a7
3	s1, s2	T0, T1, T2	a0, a2, a6, a7

The second step in multi-agent coordinating is to execute the human strategy by assignment (or allocation) of these agents to threads. We assume that the IC has sent

four free agents to the thread 1. To distribute agents among threads, the IC applies two simple methods: 1) select a sufficient subset of available agents for a thread for which this subset provides a maximum amount of capabilities and 2) select a sufficient subset of available agents for a thread for which this subset provides a minimum but sufficient amount of capabilities. The results is a set of two alternatives for selecting a choice as the final strategic decision as Table 4.VI shows this set. A choice e.g. the 1st alternative is selected by the IC, and its information is disseminated. This decision constrains all four agent to the thread 1, so agents are allowed to only do any reconnaissance or search tasks which are distributed in {s3, s4, s5}.

In this step, the IC is faced with the following questions:

- What alternatives are available for making a choice?
- How to calculate a set of feasible alternatives?
- When to calculate these alternatives?

TABLE 4.6: Two Kinds of Assignment of Agents (AoA) to the Human Strategy as Two Alternatives for Making a Choice in Time 0.

No.	**Method**	**AoA to Thread 1**	**AoA to Thread 2**	**AoA to Thread 3**
1	max	a0, a2, a6, a7		
2	min	a2	a6	a0, a7

During operations, the IC continuously monitors the real-time state of the task environment in oder to identify a right time in which the current strategic decision should be adapted. This requires the IC to continusely observe real-time and updated information of the whole environment and identify a subset of assigned agents that should be released from their threads and enter into next threads. It will result in re-running the strategic decision making procedure. The IC is faced with the following questions:

- Is it a right time to revise/adjust the strategic decision now?
- Which agents should leave which threads?
- Which thread should receive a released agent?

4.4 Literature Review

The problem addressed by this chapter is completely studied, and also a data model is designed for formulating the problem [13]. Design of a GIS-based intelligent software

system, which is called GICoordinator, has been proposed for this problem, and required capabilities of this system have been defined [14]. These works provide essential information for this chapter to design and develop any proper algorithm.

The incident commander system is a disaster management tool based on a series of rational bureaucratic principles for disaster responses [2]. Basic system objectives and plans are established at or near the top of the hierarchy and used as bases for decisions and behaviors at lower levels. Unfortunately, FEMA provides a set of useful manuals and guidelines about practices but it does not make explicit the design requirements and algorithms for making incident action plans.

STaC is a proper approach that addresses strategic planning problems [10]. There are two inefficiencies in this approach. First one is that STaC is based on the C_TAEMS that is not perfect for modeling the problem of this problem [13]. Second one is that STaC selects and automatically assigns an efficient set of agents that provides threads the minimum capabilities. This kind of decision-making sometimes is a wrong decision for team because sometimes the best decision is to keep all useful agents to a thread.

There are a few works related to the task assignment problem in a team that are applicable by an IC. A spatio-temporal task allocation algorithm to human groups is proposed for rescue operations management [18]. The goal of the RoboCupRescue simulation project is to build a simulator of rescue teams (fire-fighting, police, ambulance) acting in large urban disasters to to minimize damages caused by earthquake such as civilians buried, buildings on fire and blocked roads [8]. Unfortunately, these approaches are automated systems that do not involve human in the planning loop. In addition, decisions made by this approaches specify actions of agents fully and explicitly.

A few related approaches aimed to be mixed by human teams. A SpDI2A (Spatially Distributed Intelligent Assistant Agents) is mixed with a field human to form a human-agent team; and human-agent teams aim to distribute search and rescue tasks among themselves and assign tasks to proper teams [12]. The goal of COORDINATORS program is to create distributed intelligent software systems that help fielded units adapt their mission plans as the situation around them changes and impacts their plans; and a single COORDINATOR is partnered with each tactical unit to collaborate and coordinate with other tactical units to optimize needed mission changes [1], [9]. Unfortunately, these systems address tactical decisions that are not applicable for incident commanders for strategic planning problems.

DEFACTO incorporates artificial intelligence, 3D visualization and human-interaction reasoning into a unique high fidelity system for training incident commanders. Key aspect focuses on adjustable autonomy that refers to an agent's ability to dynamically

change its own autonomy, possibly to transfer control over a decision to a human or another agent [17]. Adjustable autonomy is different with strategic planning problems which incident commanders try to solve.

4.5 Problem Statement

To get a better insight of the problem, Fig. 4.2 presents the structure of the problem with which this chapter is faced. Required data and desirable results are shows in this figure. This section is dedicated to description of these elements.

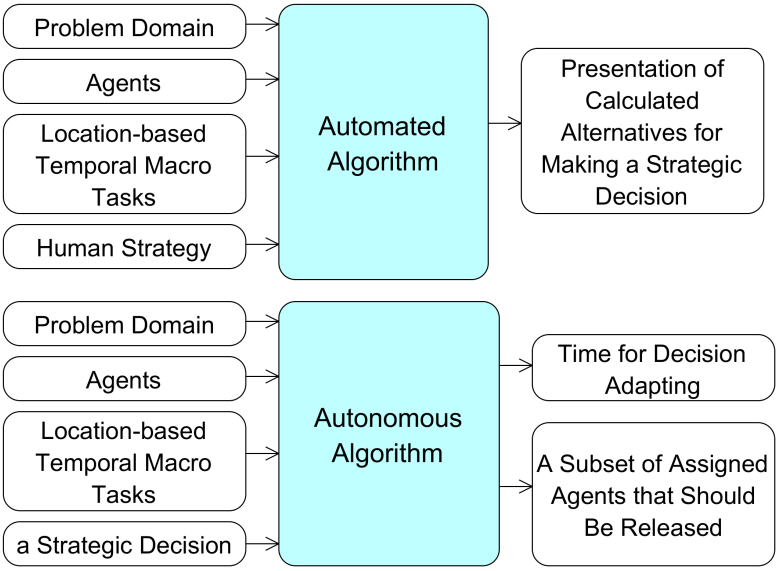


FIGURE 4.2: Structure of the problem.

4.5.1 The Problem Domain

The domain of emergency/crisis response is concerned with reducing number of fatalities in the first few days after disaster (natural or human-made), and USAR (Urban Search and Rescue) has a significant issue in this domain. USAR operations involves four task types: (1) reconnaissance and assessment by collecting information on the extent of earthquake damage, (2) search and locate victims trapped in collapsed structures, (3) extract and rescue trapped victims, and (4) transport and dispatch injured survivors to hospitals and refuges; Rescue tasks, also, are classified into three categories: light, medium, and heavy rescue according to their capabilities requirements.

Teams are often organized hierarchically that this is especially true for command and control organizations in the military and emergency response. A disaster response team

such as a firefighting team or INSARAG has a hierarchical structure which is composed of an incident commander in the top level of the team and some agents in the down node of the hierarchy.

4.5.2 Agents

In real domains, a team of agents– field units or robots – is faced with the problem of carrying out geographically dispersed tasks under evolving execution circumstances in a manner that achieves a high-level objective in a minimum time. These agents are spatially distributed in a geographic environment, have different capabilities, move from one location to another, perceive their local environment, execute strategic decisions made by their incident commander (IC), have partial information of the world‘s state, make fully decisions about their own actions, coordinate their actions with each other, cooperate with each other, perform various tasks, and report to the operations center; In addition, there is an uncertainty in actions duration and outcome of actions. Role of the IC is to coordinate and control agents because he has a global and big picture of the world's state.

Tasks that must be performed are spatially located in geographic objects (building, road segments, city blocks, or zones), require one capability or several synchronous capabilities, entail various dependencies, have dynamic and temporal quantities, and take considering time to be completed. Team members have incomplete and uncertain information of tasks (spatial distribution, quantity, and duration of tasks), and new tasks may be revealed by actions of agents or are discovered by them in time and space.

4.5.3 Location-based Temporal Macro Tasks (LoTeM)

Information of LoTeM tasks forms a global picture of the tasks environment for incident commanders. A LoTeM task is an aggregative task that aggregates a subset of tasks that has two criteria: 1) identified tasks are from a same task type and 2) theses tasks are spatially contained within a specific geographic object (or adjacent to a road segment) to which this LoTeM task is located. LoTeM tasks are encoded by geographic information.

In a definite time, a LoTeM task contains two variables: 1) an "enabled" number of tasks and 2) a "not yet enabled" number of tasks. The enabled variable states that how many available tasks are observed, discovered, or revealed within an associated geographic object. Agents can do only tasks which have become enabled. The not yet enabled amount states that how many homogeneous tasks are estimated to be revealed or discovered at future. These two amounts have dynamic and uncertain quantities that

may vary over time because some agents accomplish the enabled part while other agents may reveal the another part. These amounts give an estimation of a total duration and total capabilities which are required to do this task type in a geographic area.

There are interdependencies between LoTeM tasks which are associated with a specific location. For example, reconnaissance LoTeM task (enabled amount plus not yet enabled amount) can enable the not yet enabled part of the search LoTeM task within the same geographic area.

4.5.4 Human Strategy

Work flow of multi-agent coordination by the IC is presented in Fig. 4.3. Fist step is to define a strategy. Strategy specification enables an IC as a human to express and encode his intuition and initiative for multi-agent coordination and action planning in order to achieve a global objective. A human strategy partitions and decomposes a complex problem into a finite set of prioritized sub-problems [10], [13]. A human strategy specification is composed of a set of threads and a **thread** contains a unique ranking, a sub-team (a subset of agents), a sub-goal (a subset of task types), and sub-location (a subset of geographic areas). Human strategy can allow agents to engage in several threads.

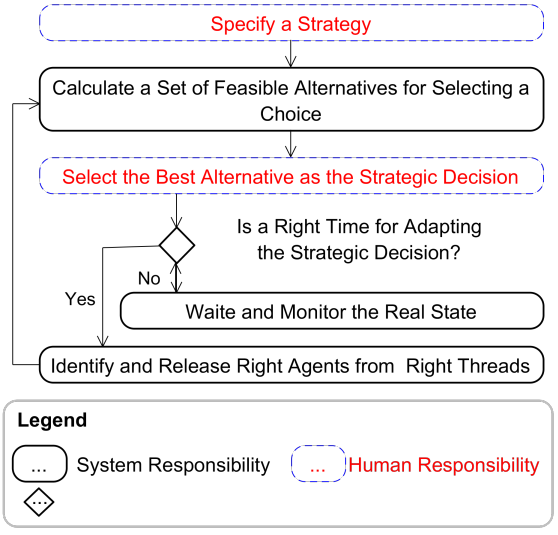


FIGURE 4.3: Work flow of multi-agent coordination by the IC.

4.5.5 Feasible Alternatives

Execution of a strategy is the strategic decision-making process. It is the problem of assignment/allocation of *available* agents to threads in an explicit time. An strategic

decision states which agents are assigned to which threads and constrains actions of agents to threads. It is obvious that an agent is allocated to only one thread. A thread receives new agents, which are called available agents, from two sources: 1) from the higher thread that has released them or 2) from the human who has directly entered them in to the thread.

A thread, as an autonomous entity, can select a sufficient subset of available agents and sends unwanted ones to the lower thread; but there may be a number of these subsets. Therefore, a set of feasible alternative scenarios may be existent that present different kinds of distribution of agents among threads. A feasible alternative can be selected as a strategic decision finally.

Agents flow from a higher thread to the lower thread through a human strategy. For example, the human strategy specified in Table 4.V includes three threads as three simple sub-problems. First, human strategy execution enters/releases e.g. the agent a0 into the thread 1. Then this agent will enter into the thread 2 after the thread 1 releases it. Finally, this agent will reach the thread 3. Consequently, it takes time for this agent to reach the thread 3.

4.5.6 Time for Decision Adapting

An agent assigned to a certain thread is kept by this thread *till* this thread releases this agent into the next thread as an idle/available agent. The IC has to timely adjust and adapt a strategic decision to new state of world. It is necessary to detect a right time for releasing right agents from right threads that this procedure results in making a new strategic decision in the new time.

4.5.7 Importance of the Problem

Calculation of feasible alternatives and adaption of a strategic decision are two important responsibilities of an IC in multi-agent coordination. They are time-consuming processes. They become harder and more challenging for the IC when the human strategy contains several interdependent threads that require the IC to continuously revise and re-make strategic decisions under rapidly dynamic situations and time pressure in decision making. As a result, this chapter focuses on these two significant issues and states two research questions:

- How to develop an automated algorithm that can calculate feasible alternatives for making a choice whenever threads receives new agents?

- How to develop an autonomous algorithm that can monitor the world's state in order to identife which agent (or a set of agents) should be released from which thread (or several threads) within a strategist decision in a right time?

4.6 Problem Formulation

To develop these algorithm, the first step is to formulate the problem. The SAP data model presents elements of the problem, properties, relationships, and interaction among these elements with regard to problem data modeling [13]. This data model is used to support development of the proposed algorithm. Fig. 4.4 shows a part of the UML class diagram of this data model that is used in this chapter.

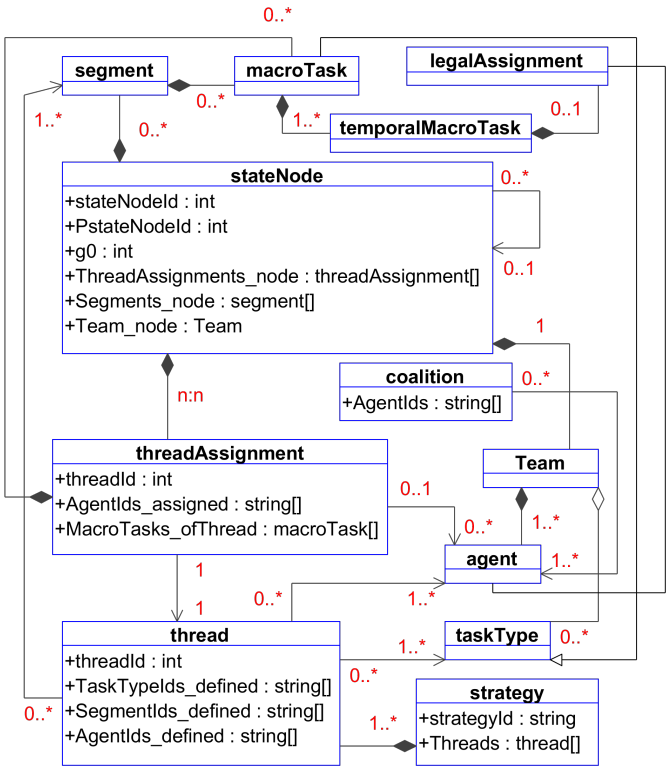


FIGURE 4.4: A part of the SAP data model that is used for the problem formulation [13].

4.7 Approach

Our approach is an automated/autonomous algorithm that assists the IC in centralized multi-agent coordination. Responsibilities of this algorithm are presented in Fig. 4.3. The proposed work flow shows a mix-initiative approach in which human and this intelligent algorithm collaborates with each other on solving a complex problem. This

algorithm includes two sub-algorithms whose pseudo codes are presented in Fig. 4.5 and Fig. 4.6.

```
Data: n :an entity of the "stateNode" class of the data model.
Data: S :an entity of the "strategy" class.
Data: D :the problem domain.
Data: p :type of selection method.
Result: N :a set of entities of the "stateNode" class that are final
        feasible alternatives.
for i ← 1 to |S.Threads| do
    t ← S.Threads[i];
    ta ← n.ThreadAssignments_node[i];
    ta.MacroTasks_ofThread ←−
    f_Calculate_MacroTasks(n.Segments_node, t, D);
end
Na ←− ∅;
Nb ←− ∅;
Nb ←− Nb ∪ {n};
for i ← 1 to |S.Threads| do
    t ← S.Threads[i];
    for nb ∈ Nb do
        ta ← nb.ThreadAssignments_node[i];
        A1 ←− f_Identify_Agents_ResidentIn(ta);
        A2 ←− f_Identify_Agents_ReceivedBy(ta);
        for m0 ∈ ta.macroTask_ofThread do
            tm0 ← m0.TemporalMacroTasks.Last();
            tm0.LegalAssignments ←−
            f_Select_Efficient_Agents(m0, t, A1, A2);
        end
        M ←− ta.macroTask_ofThread;
        C1 ←− f_Form_EfficientCoalitions(M, A1, A2);
        C2 ←− f_Purify_Coalitions(C1, M);
        C3 ←− f_Select_Coalitions(C2, p);
        for j ← 1 to |C3| do
            Na ←−
            Na ∪ {f_Generate_NewNode(C3[j], nb)};
        end
    end
    Nb ←− Na;
    Na ←− ∅;
end
N ←− Nb;
```

FIGURE 4.5: The automated algorithm for calculation of a finite set of feasible alternatives in making a strategic decision.

Algorithm presented in Fig. 4.5 is an automated algorithm that calculates a finite set of feasible alternatives in making a strategic decision. It expands a decision tree from a node in which a thread (or several threads) has received new agents from the IC or from the higher thread. Thread n of the strategy is associated with the level n of the decision tree in which it receives from the higher thread a number of nodes which are leaf nodes of the tree in fact. This thread expands the tree from all received nodes and generates new nodes for each received node. Each node expresses which agents have been assigned to higher threads and which agents have been sent into this thread. Finally new nodes generated by this thread are disseminated to the lower thread.

For each node, this thread, first, extracts LoTeM tasks associated with this thread, and then it calculates a set of efficient coalitions using all available agents (previously assigned agents and recently received agents). A efficient coalition is composed of a

Data: n :an entity of the "stateNode" class of the data model.
Data: S :an entity of the "strategy" class.
Data: D :the problem domain.
Result: $n.ThreadAssignment\_node$: the updated attribute of the $n0$.
repeat
  for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
    $t \leftarrow S.Threads[i]$;
    $ta \leftarrow n.ThreadAssignments\_node[i]$;
    $ta.MacroTasks\_ofThread \longleftarrow$
    $f\_Calculate\_MacroTasks(n.Segments\_node, t, D)$;
  end
  $A3 \longleftarrow \emptyset$;
  for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
    $t \leftarrow S.Threads[i]$;
    $ta \leftarrow n.ThreadAssignments\_node[i]$;
    $A1 \longleftarrow ta.AgentIds\_assigned$;
    for $m0 \in ta.macroTask\_ofThread$ **do**
      $tm0 \leftarrow m0.TemporalMacroTasks.Last()$;
      $A1 \longleftarrow A1 -$
      $f\_Select\_Efficient\_Agents(m0, t, A1, \{\})$;
    end
    $A3 \longleftarrow A3 \cup A1$;
  end
  if $|A3| = 0$ **then**
    $f\_WaitForaWhile()$;
  end
  else
    $TA \longleftarrow n.ThreadAssignment\_node$;
    $f\_Releas\_Agents(TA, A3)$;
    break;
  end
until *the global objective is achieved*;

FIGURE 4.6: The autonomous algorithm for adaption of a strategic decision.

subset of available agents that can provides all capabilities required by this thread with regard the strategy definition, the state of tasks, capabilities of agents, and the problem domain. A new node is generated per a coalition to show which agents are allocated to this thread and which ones are going to be released into the lower thread. It is important to remember that the state of tasks does not change and all made decisions are related to a snapshot. A node is modeled by the "stateNode" class of the data model presented in Fig. 3.4.

Algorithm presented in Fig. 4.6 is an autonomous algorithm that continuously monitors the tasks states to detect a right time in which a subset of assigned agents should be released from their threads. In real time, this algorithm continuously observes and monitors the task environment that changes over time. It aims to timely identify a subset of assigned agents that can not provide any efficient capabilities for their threads anymore. This algorithm sends these unneeded agents to a lowly prioritized thread in the identified time. A release time indicates that the strategic decision should be refined. This process results in re-executing the Algorithm of Fig. 4.5.

The following subsections are dedicated to description of sub-algorithms.

4.7.1 Calculate Macro Tasks

The purpose of Algorithm presented in Fig. 4.7 is to extract LoTeM tasks which are associated with a definite thread. This algorithms integrates the thread definition with segments-located macro tasks in order to select a subset of segment-based macro tasks that are located in the thread. It classifies the selected tasks according to the task types of the problem domain and aggregates tasks of each group to calculate a new macro task which is associated with this thread. Table 4.VII shows the achieved result form execution of this algorithm.

```
n0.Segments_node, t, D
Data: S :a set of "segment" elements.
Data: t :an entity of the "thread" class that presents a thread in
      the human strategy.
Data: D :a set of "taskType" elements in the problem domain.
Result: M : a set of "macroTask" elements that will be extracted
        for t.
begin
    M ⟵ ∅;
    for d0 ∈ D do
        m0 ← f_Initialize_aNew_macroTask();
        m0.taskType ← d0;
        m0.geoObjectId ← t.threadId;
        m0.TemporalMacroTasks ⟵
        {f_Initialize_aNew_temporalMacroTask()};
        M ⟵ M ∪ {m0};
    end
    for s0 ∈ S do
        for m0 ∈ s0.MacroTasks_ofSegment do
            if m0.taskType.taskTypeId ∈
            t.TaskTypeIds_defined and
            s0.segmentId ∈ t.segmentIds_defined then
                for i ← 1 to |M| do
                    if M[i].taskType = m0.taskType
                    then
                        c0 ←
                        |m0.TemporalMacroTasks|;
                        tm0 ←
                        m0.TemporalMacroTasks[c0];
                        tm2 ←
                        M[i].TemporalMacroTasks[0];
                        tm2.disenabledAmount ←
                        tm2.disenabledAmount +
                        tm0.disenabledAmount;
                        tm2.enabledAmount ←
                        tm2.enabledAmount +
                        tm0.enabledAmount;
                    end
                end
            end
        end
    end
end
```

FIGURE 4.7: The algorithm for extracting a set of macro tasks which is contained by a thread.

TABLE 4.7: Temporal Macro Tasks associated with the Three Threads in the Simulated USAR Scenario.

Location (Thread)	**Task-Type**	**Not Yet Enabled Amount**	**Enabled Amount**
Thread 1	Reconnaissance	0	25
Thread 1	Search	12	23
Thread 2	Light Rescue	17	16
Thread 3	Reconnaissance	0	25
Thread 3	Search	0	5
Thread 3	Light Rescue	10	9

4.7.2 Select Efficient Agents for a Macro Task

This algorithm aims to identify a subset of available agents that are efficient for a macro task of a certain thread. Two proposes are considerable: 1) define whether an available agent is allowed to be assigned to the thread and 2) calculate a maximum ability that an agent can provide for a macro task. Algorithm presented in Fig. 4.8 shows that this algorithm takes into account three criteria for selection of an agent.

Tasks require capabilities to do them and agents provide capabilities. An agent is efficient for a task if it provides any capability or capabilities required by that task. To simply this algorithm, we made an assumption that each task type requires a specific capability, while agents provide different capabilities. This assumption ignores tasks that may require simultaneous capabilities. This algorithm applies both enabled tasks and not yet enabled tasks in computation.

4.7.3 Form Efficient Coalitions for a Thread

Algorithm presented in Fig. 4.9 aims to calculate a set of efficient coalitions for a definite thread. An efficient coalition is a subset of efficient agents that can provide all capabilities required this thread. In fact, an efficient coalition has ability of doing all tasks which are contained by this thread. A coalition is a candidate whose agents will be assigned to the thread. In coalition formation theory, performance of a coalition is a function of capabilities of agents, quality of cooperation among agents, etc.

This algorithm engages all efficient and previously assigned agents in this process. If a coalition is not found, the algorithm selects a coalition that contains all efficient agents; Although this coalition does not fully provide capabilities requirements, the thread requires this coalition to do possible tasks. We assume that each macro task requires one type of capabilities.

```
Data: t :an entity of the "thread" class that presents a thread.
Data: A1 :a subset of agents assigned to t.
Data: A2 :a subset of agents released/entered into t.
Data: m : an entity of the "macroTask" class.
Result: LA :a subset of entities of the "legalAssignment" class to
        encode agents selected for the m.
LA ⟵ ∅;
tm0 ← m.TemporalMacroTasks.Last();
tm0.LegalAssignments ⟵ ∅;
for a0 ∈ A1 ∪ A2 do
    if a0.agentId ∈ t.AgentIds_defined then
        if tm0.disenabledAmount > 0 or
        tm0.enabledAmount > 0 then
            x0 ← 0;
            for C0 ∈ a0.agentType5.AgentCaps do
                c2 ← m.taskType5.CapTypeIds[0];
                n ← C0.CapTypeIds.CountOf(c2);
                if x0 < n ∗ aC0.speed then
                    x0 ← n ∗ aC0.speed;
                end
            end
            if x0 > 0 then
                lA0 ⟵
                f_Initialize_aNew_legalAssignment();
                lA0.agentId ← agent.agentId;
                lA0.maxAbility ← x0;
                LA ⟵ LA ∪ {lA0};
            end
        end
    end
end
```

FIGURE 4.8: The algorithm for selecting the efficient agents for a macro task associated with a thread.

4.7.4 Purify Coalitions

Algorithm presented in Fig. 4.10 aims to purify a set of coalitions. It is important for the system to find and eliminate redundant coalitions. The role of two coalitions, which are formed by different agents, is the same for a thread if they provide same capabilities for the thread. Redundant coalitions cause the loop problem in search algorithms.

The central part of this algorithm is an ability matrix. This matrix is used to present a matrix of total capabilities which all coalitions provide for all macro tasks. Rows are donated to macro tasks, and columns are donated to coalitions. A number in a cell shows a total capability which the relevant collation can provide for the relevant macro task. Finally, a number as the weighted total ability is calculated for each collation.

4.7.5 Select Coalitions

This simple algorithm aims to choose a number of coalitions from the purified coalitions for a specific according to a selection method. We consider two options: 1) sort the coalitions according to their weighted total ability, and then select the first one and last

```
Data: M : the set of "macroTask" elements associated with a
      specific thread.
Data: A1 :a subset of agents assigned to this thread.
Data: A2 :a subset of agents released/entered into this thread.
Result: C :a set of "coalition" elements.
begin
    C ⟵ ∅;
    C0 ⟵ ∅;
    A1 ⟵ A1 ∩ f_Read_EfficientAgents(M);
    A2 ⟵ A2 ∩ f_Read_EfficientAgents(M);
    P ⟵ f_Power(A2);
    for p0 ∈ P do
        p0 ⟵ p0 ∪ A1;
        C0 ⟵ C0 ∪ {p0};
    end
    for c0 ∈ C0 do
        uc0 ⟵ true;
        for m0 ∈ M do
            um0 ⟵ false;
            tm0 ← m0.TemporalMacroTasks.last();
            for lA0 ∈ tm0.LegalAssignments do
                if lA0.agentId ∈ c0 then
                    um0 ⟵ true ;
                    exit for;
                end
            end
            if um0 = false then
                uc0 ⟵ false;
                exit for;
            end
        end
        if uc0 = true then
            C ⟵ C ∪ {c0};
        end
    end
    if C = ∅ then
        c2 ⟵ f_Find_biggest_c0In(C0);
        C ⟵ {c2};
    end
end
```

FIGURE 4.9: The algorithm for forming a set of efficient coalitions for a thread.

one which have a minimum ability and the maximum ability and 2) select all available agents.

4.7.6 Generate a New Node

For a certain thread, a selected coalition defines which agents should be assigned to this thread and which ones have to leave the thread. A number of coalitions may be calculated, and each one presents a feasible alternative for appropriate assignment of available agents to this thread. A new node is produced per a coalition, and agents of this coalition are assigned to the related thread and unwanted agents are assigned to the next thread. This algorithm updates the "ThreadAssignments_node" attribute of the node.

```
Data: M : the set of "macroTask" elements.
Data: C :the set of "coalition" elements.
Result: C2 :a subset of C which are pured.
begin
    C2 ⟵ C;
    for c0 ← 1 to |C| do
        t0 ← 0;
        for r0 ← 1 to |M| do
            S[r0][c0] ← 0;
            m0 ⟵ M[r0];
            tm0 ← m0.TemporalMacroTasks[0];
            for lA0 ∈ tm0.LegalAssignments do
                if lA0.agentId ∈ C[c0].AgentIds then
                    S[r0][c0] ←
                    S[r0][c0] + lA0.maxAbility;
                end
            end
            t0 ← t0 + S[r0][c0];
        end
        C[c0].totalAbality ← t0 ∗ |C[c0].AgentIds|;
    end
    for c0 ← 1 to |C| do
        for c2 ← c0 + 1 to |C| do
            Eq0 ← true;
            for r0 ← 1 to |M| do
                if S[r0][c0] ≠ S[r0][c2] then
                    Eq0 ← false;
                    exit for
                end
            end
            if Eq0 = true then
                exit for
            end
        end
        if Eq0 = false then
            C2 ⟵ C2 ∪ {C[c0]};
        end
    end
end
```

FIGURE 4.10: The algorithm for purify the coalitions.

4.8 Implementation and Evaluation

The presented algorithms were implemented in development of two capabilities of GICoordinator [14]. These two algorithm were executed for the scenario which was stated in Section II in order to get a better understand of the algorithms. In addition, we tested the automated algorithm on various human strategies and different number of agents in order to calculate run time and number of generated nodes. Results are shown in the next section.

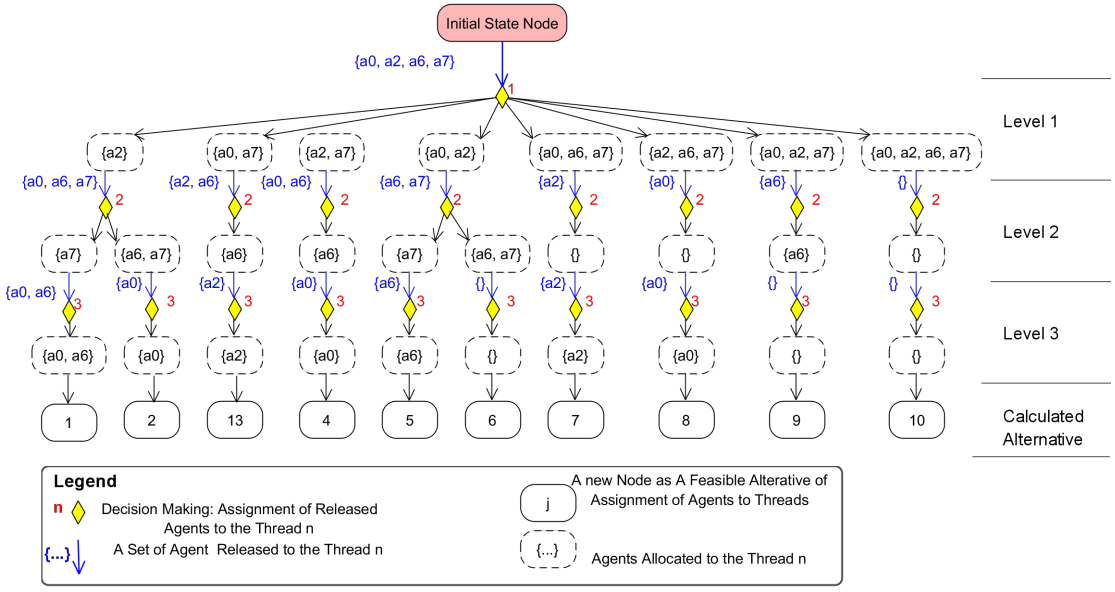


FIGURE 4.11: Expand the the decision Tree from the State Node 0 by the automated algorithm.

4.9 Result

4.9.1 Automated Calculation of Feasible Alternatives

Fig. 4.11 presents how the automated algorithm has expanded/generated a decision tree using a state node in which four free agents have been sent to the thread 1 by the IC in time 0 at the stated USAR scenario. Table 4.VIII shows the final result of this algorithm that it is a collection of 10 alternative scenarios that are presented for the IC in order to support and assist him in selecting a choice for strategic decision making in coordination of agents. 10 new nodes were generated and each one has a different assignment of agents to threads.

TABLE 4.8: A Set of Alternatives, which is Calculated by the Automated Algorithm, for Making a Choice for Strategic Decision Making in Time 0.

Choice No.	Thread Assignment 1	Thread Assignment 2	Thread Assignment 3
1	a2	a7	a0, a6
2	a2	a6, a7	a0
3	a0, a7	a6	a2
4	a2, a7	a6	a0
5	a0, a2	a7	a6
6	a0, a2	a6, a7	
7	a0, a2, a6		a2
8	a2, a6, a7		a0
9	a0, a2, a7	a6	
10	a0, a2, a6, a7		

4.9.2 Autonomous Adaption of a Strategic Decision

Imagine that 6th alternative was selected by the IC or by an intelligent assistant system. Table 4.IX shows this final strategic decision which was made in time 0. Information of this decision was sent to four agents, which are distributed in the operational area. According to this decision, actions of agents {a0, a2} will to do only reconnaissance tasks and search tasks that are spatially located/distributed at proximity of three roads {s3, s4, s5}. Also according to this decision, actions of agents {a6, a7} will to do only light rescue tasks located at the same geographic area. This mission started at time 0.

We simulated the problem by a suitable approach [15]. An proper algorithm was used to dynamically assign LoTeM tasks to agents in multi-agent scheduling.

TABLE 4.9: First Strategic Decision, which is the 6th Alternative in Time 0, for Coordination and Control of Agents.

Thread Id	**Assigned Agents**	**Start Time**
1	a0, a2	0
2	a6, a7	0
3		0

Imagine that during emergency response operations, new data are reported to the operation center from the agents. This center gathers and integrates data to continuously/temporally provide a timely situational awareness as a big picture for the IC. The IC is required to adapt the current strategic decision to new situation in a right time, therefore he has to scan and monitor the world's state. In time 96, Table 4.X shows updated information of the LoTem tasks.

TABLE 4.10: Update State of Twelve Segment-based Temporal Macro Tasks in Time 96 using the GICoordinator [15].

No.	**Location (Road S.)**	**Task-Type**	**Not Yet Enabled Amount**	**Enabled Amount**
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	*0*
6	s3	T1	*0*	*8*
7	s3	T2	2	*5*
8	s5	T0	0	*0*
9	s5	T1	*0*	*7*
10	s5	T2	5	*3*
11	s4	T1	0	*9*
12	s4	T2	*5*	*7*

The autonomous algorithm revealed that time 96 is a right time for adaption of the strategic decision. Because according to the strategic decision and the Table 4.X, there is no available task that the agent {a0} can do but other agents have to continue their jobs. The result was to release the agent {a0} from the thread 1 and send it into thread 2.

This action re-triggers the automated algorithms that the result was only one alternative as Fig. 4.12 shows.

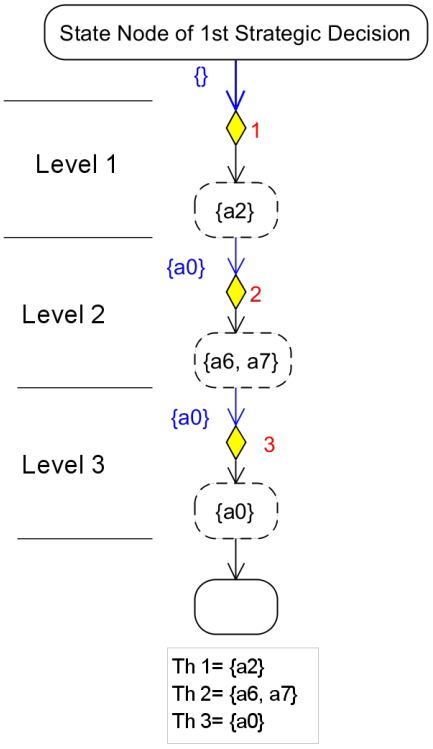


FIGURE 4.12: Autonomous adaption of the first strategic decision in Time 96 by Releasing the agent "a0" from thread 1 into the thread 2, and automated calculation of Feasible Alternative for Adaption this decision.

4.9.3 Evaluation of Efficiency of the Automated Algorithm

To evaluate running time of the automated algorithm, we executed it on the same tasks with three human strategies and different team sizes. The strategy I includes three threads that are independent in agents. The strategy II includes three semi independent threads. The strategy III which is a complex strategy is composed of three threads that all agents are defined for all threads.

Table 4.XI shows results of this text. The result achieved for a scenario scenario includes two parameters: 1) the amount of running time and 2) number of alternative which were calculated for this scenario.

..

TABLE 4.11: Evaluation of the Running Time of the Automated Algorithm.

Team		**All Coalitions**		**Min-Max Coalitions**	
Size	**Strategy**	**Run Time**	**Nodes**	**Run Time**	**Nodes**
4	I	27	1	31	1
8	I	44	16	38	8
12	I	51	81	43	8
4	II	29	2	28	2
8	II	42	28	41	6
12	II	101	171	41	6
4	III	51	10	38	3
8	III	106	172	40	4
12	III	922	1548	117	4

4.10 Conclusion

This chapter presented an intelligent algorithm composed of two sub-algorithms for 1) automated calculation of feasible alternative scenarios for selecting a strategic decision in certain time and 2) autonomously detecting a right time in which this strategic decision should be refined by releasing a set of identified agent from their threads. In fact, this algorithm autonomously controls which agents should be released from their thread within a strategic decision. The role of human is to select a alternative as a strategic decision.

A strategic decision constrains agents to threads but it does not assign tasks to agents. A strategic decision controls the domain of activities of agents.

A coalition, which is a set of agents, is sufficient for a thread if it provides all capabilities required by all tasks (enabled tasks and not yet enabled tasks) located in this thread. A question that arises here is that which coalition among available coalitions can accomplish this thread faster than others? This question will be addressed a future work.

A huge number of feasible alternatives are found for a case which contains a team with big size and a complex strategy. Selection of two coalitions that provide minimum or maximum capabilities for a thread reduce the run time and number of generated alternatives.

Importance of this algorithm is to assist human and collaborate with him in the flow chart of centralized multi-agent coordination. This issue is so important and essential especially in a situation that the human strategy defines agents for several threads.

This algorithm can be used in the multi-agent planning problem. A action plan made by the IC is a sequence of strategic decisions that states how a team of agents can achieve the goal.

The future will be an search-based algorithm that aims to make a optimal decision by selection of the best choice among available alternatives.

References

[1] Barbulescu, Laura, Zachary B. Rubinstein, Stephen F. Smith, and Terry L. Zimmerman. "Distributed coordination of mobile agent teams: the advantage of planning ahead." In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1, pp. 1331-1338. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[2] Bigley, Gregory A., and Karlene H. Roberts. "The incident command system: High-reliability organizing for complex and volatile task environments." Academy of Management Journal 44, no. 6 (2001): 1281-1299.

[3] Burstein, Mark H., and Drew V. McDermott. "Issues in the development of human-computer mixed-initiative planning." Advances in Psychology 113 (1996): 285-303.

[4] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[5] Fiedrich, Frank, Fritz Gehbauer, and U. Rickers. "Optimized resource allocation for emergency response after earthquake disasters." Safety Science 35, no. 1 (2000): 41-57.

[6] Hunger, J. David, and Thomas L. Wheelen. Essentials of strategic management. New Jersey: Prentice Hall, 2003.

[7] INSARAG Guidelines and Methodology Manual, United Nations Office for the Coordination of Humanitarian Affairs, 2004, http://www.usar.nl/upload/docs/insarag_guidelines_july_2006.pdf (accessed July 2013).

[8] Kitano, Hiroaki, and Satoshi Tadokoro. "Robocup rescue: A grand challenge for multiagent and intelligent systems." AI Magazine 22, no. 1 (2001): 39.

[9] Maheswaran, Rajiv T., Pedro Szekely, Marcel Becker, Stephen Fitzpatrick, Gergely Gati, Jing Jin, Robert Neches et al. "Predictability & criticality metrics for coordination in complex environments." In Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2, pp. 647-654. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[10] Maheswaran, Rajiv T., Pedro Szekely, and Romeo Sanchez. "Automated adaptation of strategic guidance in multiagent coordination." In Agents in Principle, Agents in Practice, pp. 247-262. Springer Berlin Heidelberg, 2011.

[11] Malone, Thomas W., and Kevin Crowston. "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) 26, no. 1 (1994): 87-119.

[12]

[13] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research (Manuscript submitted for publication), 2013.

[14] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." In Workshop on Robots and Sensors integration in future rescue INformation system, ROSIN'13. In conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), 2013.

[15] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC). (in press)

[16] Nwana, Hyacinth S., Lyndon C. Lee, and Nicholas R. Jennings. "Coordination in software agent systems." British Telecom Technical Journal 14, no. 4 (1996): 79-88.

[17] Schurr, Nathan, Janusz Marecki, John P. Lewis, Milind Tambe, and Paul Scerri. "The defacto system: Training tool for incident commanders." In AAAI, pp. 1555-1562. 2005.

[18] Vafaeinezhad, Ali Reza, Ali Asghar Alesheikh, Majid Hamrah, Reza Nourjou, and Rouzbeh Shad. "Using GIS to Develop an Efficient Spatio-temporal Task

Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications–ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009. Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

Chapter 5

Automated Algorithm for Assignment of Location-based Temporal Macro Tasks to Field Units

5.1 Abstract

This chapter addresses a centralized scheduling problem in multi-agent systems by which the incident commander (I.C.) of a disaster response team aims to coordinate actions of field units (rational agents) in order to minimize the total operation time in uncertain, dynamic, and spatial environments. The purpose of this chapter is to propose an intelligent software system that assists the I.C. in dynamic assignment of geospatial-temporal macro tasks to agents under human strategic decisions. This system autonomously executes a heuristic algorithm to minimize the maximum total dependent duration according to human high-level strategies. The result is a schedule composed of macro decisions that each one states seven types of information: 1) what task type is going to be done, 2) who (a subset of agents) are assigned to do this assignment, 3) where (a road segment or zone as a macro geospatial object) contains a subset of tasks, 4) when operations start, 5) when operations finish, 6) how many tasks are estimated to be done, 7) what task types and how many of them are estimated to be revealed in this location after to finish this job. This result, which is a feasible solution for the addressed problem, permits the I.C. to coordinate agents, partially specify activities of agents in time and space, minimize the overall execution time for all tasks, calculate a

right time to revise a strategic decision, evaluate the efficiency of a high-level strategy, and estimate the makespan.

5.2 Introduction

The objective of scheduling is to minimize the overall execution time for all tasks by properly allocating tasks to the processors/machines without violating the precedence constraints among the tasks [9]. The input of these problems is usually considered as a directed acyclic graph (DAG) which provides precedence, dependency, priority among tasks, cost, and other information for tasks [8]. The total run-time is called makespan in literature.

Urban search and rescue (USAR) are considered the major function of disaster emergency response operations. Its objective is to reduce number of fatalities in the first few days after disaster [5]. The scheduling problem is an important and crucial issue for incident commanders who try to coordinate activities of field units (rational agents) during disaster crisis/emergency response management. It aims to specify and schedule actions of agents by appropriately assignment/allocation of tasks to agents. Although scheduling enables agents to accomplish tasks in the minimum overall time, it is difficult and hard to solve this problem under uncertain, spatial, and dynamic situations.

A responder team such as INSARAG (International Search and Rescue Advisory Group) has a hierarchical structure consisting of two levels. The down level in the hierarchy includes rational agents. They are spatially distributed in a geographic environment, perceive their local environment, execute strategic decisions made by the top node, rationally make decisions about their own actions, coordinate their actions with each other in an distributed approach, do tasks, and report to the operation center [12]. An I.C. who is located at the top node has a global and timely view/picture of whole crisis situation. His role is to coordinate and command agents by strategic planning and scheduling in a centralized approach. As a result, a team is a society of cooperative agents that try to maximize the global utility of the team.

Coordination is the act of managing interdependencies among activities [11]. Coordination in disaster emergency response includes management of task flow, recourse, information, decision, and responder [3]. This chapter recognizes the importance of five main types of coordination in a disaster response team:

1. **Tasks Dependencies**: The *Enabling* dependency between tasks specifies that when a task is done completely, it makes possibility of performing other dependent

tasks. In other words, time that a disenabled task gets enabled is dependent to the finishing time of a task that makes it revealed.

2. **Actions Dependencies**: Heterogeneous agents can do tasks which have been revealed by actions of other agents before. Some agents may wait till their tasks get released by others.

3. **Redundant Actions**: Agents may posses overlapping capabilities that allow them to be involved in doing same tasks. It causes conflict between actions or redundant actions.

4. **Information Sharing**: Information sharing lets agents know 1) what state tasks have, 2) what tasks are going to be done by who and when, and 3) what tasks are estimated to be revealed when and where. These data enable a team to coordinate agents and adapt multi-agent scheduling.

5. **Agents Allocation**: Agents are considered as limited resources or machines which should be optimally scheduled and allocated to tasks.

However many approaches have been developed for scheduling problems, some challenges have remain at the problem addressed by this chapter. We categorize these limitations and deficiencies as follows:

- These approaches explicitly and fully schedule and specify actions of rational and autonomous agents who are located in uncertain and dynamic environments. They try to associate a certain task with a certain agent. They do not provide an efficient approach for an I.C. because situational awareness which an I.C. has from the global picture is different with perception which agents have from their local surrounding environment.

- They are automated systems that do not involve human (the I.C.) in the loop. The main obstacle is scale as it is currently infeasible for a fully automated system to effectively reason about all the possible futures that may arise during execution of tasks in a complex environment.

- They do not address scheduling problems in which macro tasks are spatially distributed in a complex environment. In addition, geospatial reasoning and geoinformation analyses play a main role in problem solving. Spatiotemporal macro tasks and characteristics of this environment make the assignment problem harder.

This chapter addresses the capability of GICoordinator in macro action scheduling. GICoordinator is a GIS-based intelligent software system that collaborates with human

and assists the I.C. in action coordination of agents in complex situations such as disaster emergency response [14]. This system autonomously executes an algorithm for assignment of spatiotemporal macro tasks to rational agents under human strategic decisions in a spatial and dynamic environment. The goal of this algorithm to minimize the longest total dependent duration by dynamic assignment of idle agents to these tasks.

The contribution of this chapter is the heuristic algorithm that calculates a feasible solution for the addressed problem. This algorithm enables us to develop this assistant software agent that can collaborate and cooperate with an I.C. on scheduling in multiage systems for multiagent coordination. It calculates a set of macro (semi-strategic) decisions that temporally constrain actions of agents according to the human strategy.

5.3 Literature Review

The scheduling problem for multiprocessor systems is stated as "How can we execute a set of tasks T on a set of processors P subject to some set of optimizing criteria C?" A taxonomy of scheduling algorithms in parallel systems and distributed systems was presented by [1]. Some algorithms have been developed to solve this problem [9, 20].

The algorithm presented by [7] aims to minimize maximum lateness and total delay for multi-machine problems concerned with the scheduling of single-operation jobs. Several heuristic algorithms were proposed by [4, 16] for assignment of n independent tasks to m unrelated parallel machines to minimize the maximum completion time. This approach was extended by this chapter to propose a new algorithm for the addressed problem.

To assign spatially distributed tasks to field units (agents) in USAR management, some algorithms were suggested by [12, 19]. These algorithms involved geographic information and geospatial analyses in properly assignment of tasks to agents.

There are several mechanisms, techniques, and algorithms for tackling task assignment/assignments to agents (human, robot, or intelligent system). They help agents to achieve their objectives and to maximize the benefits of the system. Some works address the tasks scheduling problems in multi-agent systems [10, 15], multi-robot systems[6], disaster emergency teams [18], and robocup rescue simulation [2, 17].

5.4 Problem Statement

The problem addressed by this chapter has characteristics which the following subsections describe.

5.4.1 Problem Domain

USAR has a significant issue in earthquake disaster response. The global goal is to rescue the greatest number of people who are trapped under the debris of damaged buildings in the shortest amount of time. USAR tasks involves *a sequence of dependent* tasks: (1) reconnaissance and assessment by collecting information on the extent of earthquake damage, (2) search and locate victims trapped in collapsed structures, and (3) extract and rescue trapped victims, and (4) transport and dispatch injured survivors to hospitals or refuges. Rescue tasks are classified into light rescue, medium rescue, and heavy rescue.

USAR tasks are location-based entities that are distributed in a geographical area. A specific task needs a specific capability or several synchronous capabilities to carry out this task in a considering duration. Capability requirements determine what agents are allowed to do what tasks. There is an uncertainty in tasks duration, distribution of tasks, and outcome of tasks.

5.4.2 Agents

Field units are considered mobile, spatial, rational, heterogeneous, and semi-autonomous agents. Their main roles are to do tasks using their capabilities. They possess different capabilities that allow them to engage in doing tasks for which they can provide required capability. Moreover, agents execute their capabilities with different speed. There is an uncertainty in actions duration, speed, and outcome of actions.

The I.C. as a planning human tries to coordinate disaster response management. He or she has inaccessible, global, and uncertain information about the environments state. His perception/observation of disaster situation is different with agents perception of their local environment.

5.4.3 Macro Tasks

Macro tasks information forms a global picture of the tasks environment for incident commanders. A macro task is the aggregation of all tasks that 1) are from a same task type and 2) are spatially contained within a specific geographic object or adjacent to a road segment in a definite time. Macro tasks are encoded by road segments in this chapter. Road segment are surrounded by buildings that contain USAR tasks, so a segment contains a number of macro tasks which are located to this geographic object.

A macro task contains two variables: 1) an enabled amount of tasks and 2) a disenabled amount of tasks. The enabled variable states that how many homogeneous tasks are observed, discovered, or revealed within an associated geo-object. Agents can do only tasks which are enabled . The disenabled part states that how many homogeneous tasks are estimated to be revealed or discovered at future. They have dynamic and uncertain quantities that may vary over time because some agents accomplish the enabled part while other agents may reveal the another part. These amounts give an estimation of a total duration and total capabilities which are required to do this macro task.

There is a complex interdependency relationship among variables of macro tasks which are associated with a geo-object. For example, reconnaissance macro tasks (enabled amount plus disenabled amount) can reveal only the disenabled part of a search macro task.

5.4.4 Human Strategic Decisions

A high-level strategy specification enables an I.C. to express and encode his intuition and initiative for multi-agent planning problems solving. A strategy partitions and decomposes a complex problem into a set of small problems under human supervision. A strategy contains a set of parallel, interdependent, and prioritized threads. A thread is a subproblem that is composed of a unique ranking, a subset of agents, a subset of task types, and a subset of geographic area.

Strategic decision-making is the problem of distribution and allocation of agents among threads in an explicit time. Because agents may be shared among threads, an agent should be assigned to only one thread in a time; and this agent will be available for the next thread whenever its thread does not need it and releases it. A human strategic decision defines which agents are assigned to which threads in an explicit time. A thread constrains and limits actions of assigned agents according to its specification. A strategic decision is refined and adapted to new states of the world.

5.4.5 Scheduling in Multi-agent Systems

Scheduling is the problem of assignment of macro tasks to rational agents under human strategic decisions in multi-agent systems. It does not explicitly specify actions of agents but it partially specifies actions and dynamically constrains activities and behaviors of agents in time and space. It allows rational agents to make and adapt their own decisions according to calculated scheduling.

Scheduling has a macro characteristic. More than one agent can be assigned to a macro task simultaneously. Assigned agents form a coalition that will execute this assignment cooperatively.

Assignments are dynamic. Over time, new agents can be assigned to a macro task to which a set of agents have been allocated before.

Macro tasks keep assigned agents till enabled tasks are accomplished completely. Released agents can be assigned to other macro tasks.

The objective is to minimize the overall time for USAR tasks execution.

Because of spatial distribution of tasks, agents need a travel time to reach macro tasks by moving from one location to another via a road network. In addition, streets have varying states (blocked or cleaned) that affect shortest paths.

A coalition formed by many and professional agents can do a macro task faster than another coalition. An efficient coalition has many capabilities that can finish a macro task faster than another coalition.

Action scheduling is calculated with regard to a human strategic decision till this strategic decision is valid. It consists of a sequence of temporal scheduling.

Scheduling changes states of macro tasks. An assignment can define how much of a macro task is scheduled to be done, or it can define how much of another macro task will be revealed after accomplishment.

5.5 Approach

The proposed approach which is shown in Fig. 5.1 is used by incident commanders for multiagent coordination. The focus of this chapter is on the 4th and 7th components.

Whenever a human strategic decision, this system autonomously executes a heuristic algorithm for dynamic assignment of spatiotemporal macro tasks to rational agents under human strategic decisions for scheduling problem solving. This algorithm, which is shown in Algorithm 1, is composed of 1) select efficient agents, 2) select active macro-tasks, 3) identify the release time, 4) select idle agents, 5) nominate macro task, 6), calculate utilities, 7) find the highest utilities, 8) assign agents to macro tasks, and 9) calculate the earliest finish time.

This chapter applied the SAP data model [13] for presentation and formulation the problem. It enables us to design, implement, and develop the proposed algorithm. Fig. 5.2 shows a part of this data model that presents some important information classes.

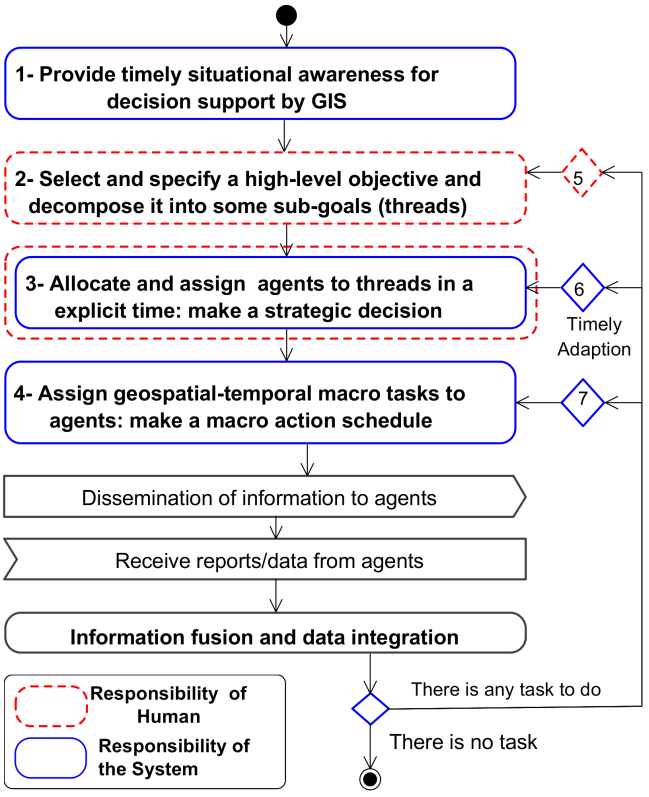


FIGURE 5.1: Role of the central scheduling problem in an approach proposed for strategic planning and scheduling in disaster response team [13].

5.5.1 Select Efficient Agents

This algorithm aims to select efficient agents for macro tasks. It is intended for two proposes. First one is to identify which agents can be assigned to which macro tasks. Second one is to calculate a maximum ability which an assigned agent can provide for a macro task. Agent selection for macro tasks is important because a selected agent states that this agent is a useful candidate who can be legally assigned to this macro and get involved in doing tasks. This algorithm takes into account four key criteria to select efficient agents for a definite macro task:

1. Does this macro task contain any task?
2. Does the human strategic decision allow this agent to be allocated to the location of this macro task?
3. Does the human strategic decision allow this agent to be assigned to the task type of this macro task?
4. Does this agent have any useful capability which is required by this macro task? And what maximum capability is provide by this agent?

An agent can be selected for more than one macro task, and more than one agent can be selected for a macro task. After agent selection for a definite macro task, this algorithm creates a set of *legalAssignment* elements for this macro task. A macro task may contain a number of legalAssignment elements that express which agents are allowed to be assigned to this macro task. A legal assignment is composed of three kinds of

```
Data: startTime : a time in which a new strategic decision is made by the incident
      commander. It triggers the scheduling algorithm.
g2 ← startTime;
L ⟵ Create_emptyset_of_legalAssignment();
while true do
|Select_Efficient_Agents();
|T ⟵ Select_Active_MacroTasks();
|A ⟵ Select_Idle_Agents(g2);
|L ⟵ L ∪ A;
|if Identify_theRelease_Time() = true then
|return;
|end
|while true do
|if |L| = 0 or |T| = 0 then
|break;
|end
|T2 ⟵ Nominate_MacroTasks();
|U ⟵ Calculate_Utilities();
|u ⟵ Find_theHighest_Utilities();
|if u.benefitRation >= 10 then
|Assign_Agents_toMacroTasks();
|continue;
|end
|else
|T ⟵ T − T2;
|continue;
|end
|end
|if |L| = 0 and |T| > 0 then
|g2 ← Calculate_earliestFinishTime();
|continue;
|end
|else if |L| > 0 and |T| = 0 then
|if Identify_theRelease_Time() = true then
|return;
|end
|else
|g2 ← Update_ProblemState();
|if g2 = −99 then
|return;
|end
|else
|continue;
|end
|end
|end
|end
```

Algorithm 1: The heuristic algorithm for dynamic assignment of spatiotemporal macro tasks to rational agents under human strategic decisions for centralized scheduling in multiagent systems

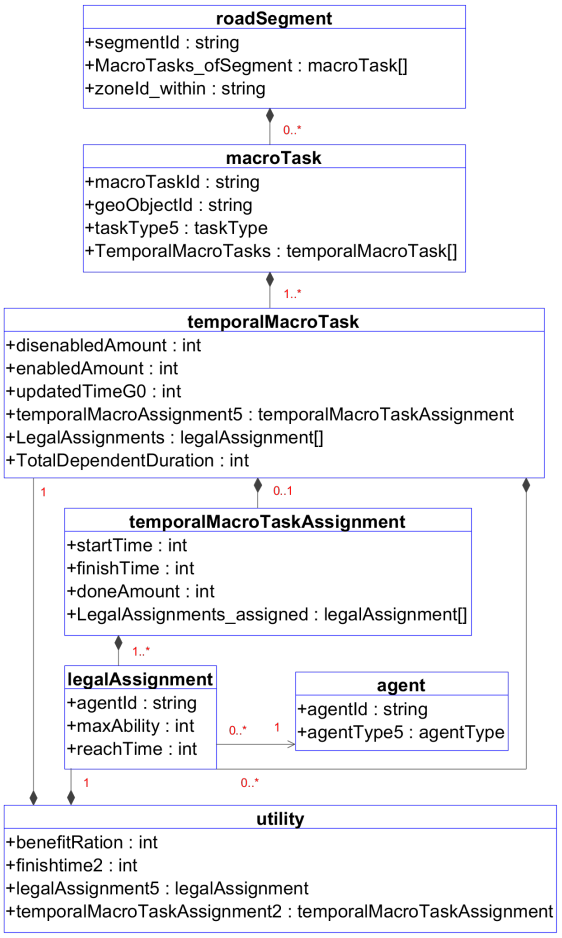


FIGURE 5.2: A part of SAP data model that includes some important information classes for this chapter.

information: 1) Id of an agent, 2) a maximum capability which the associated agent provides with regard to capability requirements of this macro task, and 3) an estimated time at which this agent will arrive in the location of this macro task and will be available for it in order to start doing tasks.

An agent is efficient for a task if it provides any capability or capabilities required by that task under the human strategic decisions. To simply this algorithm, we made an assumption that each task type requires a specific capability, while agents provide different capabilities. This assumption ignores tasks that may require simultaneous capabilities. Another considerable matter in this algorithm is to use not only the enabled tasks but also disenabled tasks. These tasks are estimated and expected to be revealed, enabled, or discovered by other tasks at future. Because of the dynamic environment, this algorithm enables the system to automatically and timely recognize agents who are not efficient anymore by macro tasks.

5.5.2 Select Active Macro Tasks

The aim of this algorithm is to select active macro tasks from the tasks environment. It is important to identify and recognize only a subset of macro tasks to which agents can be potentially allocated. Agents can accomplish only a part of macro tasks which are enabled. If agents are assigned to disenabled tasks, they get inactive, so they have to wait till tasks get discovered, revealed, or enabled by other tasks.

This algorithm applies two criteria to identify a macro task as an active one:

1. Does this macro task contain any enabled amount of tasks?
2. Does this macro task contain any "legalAssignment"?

5.5.3 Select Idle Agents

This algorithm aims to select idle agents from the team at the current time "g2". Selected agents are candidates who will be assigned to active macro tasks appropriately.

Two sources dynamically and temporally provide idles agents for this algorithm:

- Incident commander who defines and enters new agents in the team.
- Macro tasks that release their own assigned agents.

A macro task releases its agents who are expected to do enabled tasks by the time g2. This algorithm accesses information of macro tasks and agent allocation, so it gathers all idle agents which are released by some macro tasks at this time. A "legalAssignment" element is used to encode a released agent and present a last location (Id of a road segment) in where the agent is. Finally, the algorithm adds selected agents to the current collection called L in the algorithm.

5.5.4 Identify the Release Time

This algorithm determines that whether it is a right decision to release a subset of idle agents right now at time "g2". It is import to timely adapt a strategic decision to new situation by releasing unneeded agents from threads and sending them to other threads that may require them. The algorithm helps the system to select a choice from two ones: 1) keep on scheduling with all idle agents under the same strategic decision or 2) stop scheduling and let the system revise the current strategic decision.

As Algorithm 2 shows, this algorithm takes into account two criteria to keep an idle agent for task assignment:

- Is there any macro task for which this agent has been categorized as an efficient agent before?
- Is it estimated that an identified macro task leaves any task (enabled tasks or disenabled tasks) at the current time?

5.5.5 Nominate Macro Tasks

This heuristic algorithm aims to select a subset of active macro tasks. Idle agents are going to be assigned to the nominated macro tasks. This algorithm versus search algorithms selects the best node and extends it to reach the goal node in which all tasks are finished.

The logic of this algorithm is to select macro tasks that have a strong effect on minimizing the makespan. This algorithm nominates macro tasks that have a **longest** "total dependent duration". Two sources of information are important for calculation this dynamic variable for each macro task for the current time:

Data: Lf :a set of "legalAssignment" elements that express idle agents.
Data: S :a set of "segment" elements that contain macro tasks.
Result: b :a boolean variable to show whether it is a right time to release a subset of idle agents.

$Ls \longleftarrow Lf$;
for $l \in Lf$ **do**
  for $s \in S$ **do**
  for $m \in s.MacroTasks\_ofSegment$ **do**
  $c \leftarrow |m.TemporalMacroTasks|$;
  $t \longleftarrow m.TemporalMacroTasks[c]$;
  $g \longleftarrow t.temporalMacroAssignment$;
  if $l \in t.LegalAssignments$ **then**
  $n \leftarrow t.disenabledAmount + t.enabledAmount - g.doneAmount$;
  if $n > 0$ **then**
  $Ls \longleftarrow Ls - \{l\}$;
  end
  end
  end
  end
end
if $|Ls| > 0$ **then**
  $b \leftarrow true$;
end
else
  $b \leftarrow false$;
end

Algorithm 2: the "Identify_theRelease_Time()" algorithm that determines whether it is a right decision to release a subset of idle agents right now

- How many enabled tasks are contained in this macro task? or when does the enabled part of this macro task get done according to the scheduling which has been calculated before?
- How many dependent tasks are estimated proximately that this macro task can reveal?

5.5.6 Calculate Utilities

This algorithm aims to calculate an utility matrix of idle agents for the nominated macro tasks. It is important to answer this question: How much can a macro task benefit from allocation of a specific agent? This matrix assists us to select and allocate the best agents to macro tasks. A benefit states that how much an agent will reduce the finish time of a macro task, if this agent is assigned to this task. This algorithm takes into account several factors in calculation of the benefit of an definite agent for a definite macro task:

1. Is it legal for this idle agent to be assigned to this nominated macro task?
2. When will this agent start doing tasks? The dynamic amount of this time is dependent to three factors: 1) moving speed of this agent, 2) release time of this agent, and 3) travel time and distance of between two locations in the road network.
3. What capability does this agent provide for this macro task? With what speed can this agent carry out tasks?
4. How much can this agent improve performance of a coalition which has been assigned to macro task before?

This utility matrix is used to present a matrix of **utility** elements which idle agents provide for nominated macro tasks. Rows are donated to macro tasks, and columns are donated to agents. A set of numbers in a cell shows information of an utility which the relevant agent can provide for the relevant macro task.

5.5.7 Find the Highest Utilities

The goal of this algorithm is to determine that whether there is a subset of idle agents who can provide a benefit-ratio higher than 10%. This algorithm uses the utility matrix to find these agents who provide acceptable benefits for nominated macro tasks. These agents will be assigned to related macro tasks. If a suitable agent is not found, all nominated macro tasks will be removed by the algorithm.

5.5.8 Assign Agents to Macro Tasks

This heuristic algorithm shown in Algorithm 3. It aims to select proper agents and assign them to macro tasks. This algorithm uses the utility matrix and applies a minimax algorithm to select the best agents for the best macro tasks. This algorithm selects all agents that decrease the finish time of macro tasks more than 10%. If an agent is allocated to a macro task, the associated row and column of the matrix is removed, and also the assigned agent is removed from the list of idle agents.

5.5.9 Calculate the Earliest Finish Time

This algorithm aims to calculate the earliest time in which some agents will get released and idle by some macro tasks. It updates the problem state that includes agents, tasks, scheduling, finish time of each macro task, etc. This algorithm enables us to simulate the problem and estimate future states of the world. A new subset of macro tasks is identified that will be completed sooner than others. Results let other algorithms continue scheduling.

Data: TU : a set of macroTask-Utilities that a member includes a set of "utility" elements for a nominated macro task.
Data: Lf :a set of "legalAssignment" elements that express idle agents.
while $|TU| > 0$ **do**
$MiN \longleftarrow \emptyset$;
for $tU \in TU$ **do**
$U2 \longleftarrow Find\_theHighest\_Utilities\_in\_tU(10)$;
if $|U2| = 0$ **then**
$MiN \longleftarrow MiN + \{0\}$;
end
else
$MiN \longleftarrow MiN + \{Select\_theMin\_FinishTime\_in\_U2()\}$;
end
end
$m \leftarrow Select\_theMax\_Number\_in\_MiN()$;
if $m = 0$ **then**
break loop;
end
$u \longleftarrow Find\_theUtility(m)$;
$t \longleftarrow Find\_theMacroTask(m)$;
$a \longleftarrow Find\_theAgent(m)$;
$n \leftarrow |t.TemporalMacroTasks|$;
$t[n].temporalMacroAssignment5 \longleftarrow CreateUpdate\_with\_u()$;
$Lf \longleftarrow Lf - \{a\}$;
$TU \longleftarrow TU - \{Find\_tU(t)\}$;
for $tU \in TU$ **do**
$tU \longleftarrow tU - \{u\}$;
end
end

Algorithm 3: Agent selection for assignment of tasks to agents

TABLE 5.1: A human high-level strategic decision which is valid from time 0 and is used for scheduling.

thread Id	a set of zones	a set of task types	a set of agents
1	zo1, zo2	T0-Reconnaissance	a0, a1, a2, a3, a4, a5
2	zo1, zo2	T1-Search	a7
3	zo1, zo2	T2-SlightRescue, T3-MediumRescue, T4-HeavyRescue	a9, a11
4	zo3, zo4	T0-Reconnaissance, T1-Search, T2-SlightRescue, T3-MediumRescue, T4-HeavyRescue	a6, a8, a10

5.6 Evaluation and Results

To evaluate the proposed approach, it was used for an USAR scenario simulated by GIS. As Fig. 5.3 shows, a rapid response team which is composed of an I.C. and 12 agents was assigned to the operational area. The global objective was to accomplish USAR tasks by this team in a minimum time. Table 5.1 shows a strategic decision which is used for scheduling.

Two main results are achieved by running the proposed algorithm: 1) a feasible schedule

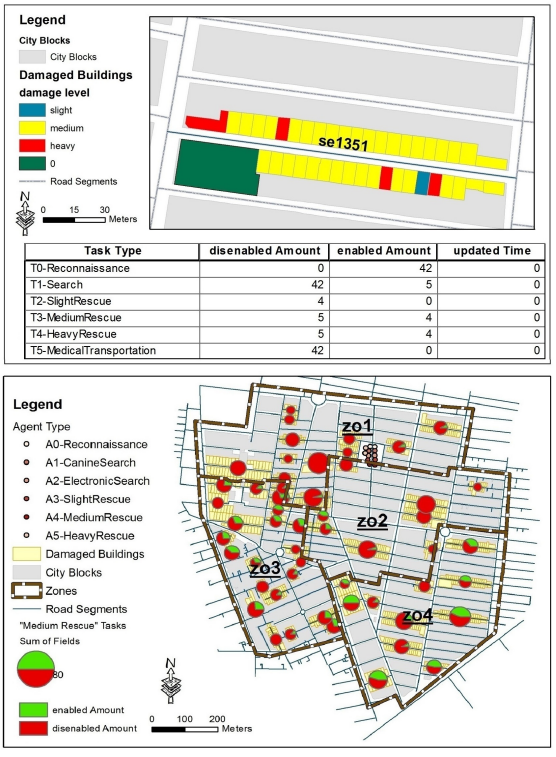


FIGURE 5.3: Two thematic maps of spatial distribution of macro tasks.

and 2) an adaption/refinement time. In addition, two minor results can be achieved: the makespan, a total schedule.

A schedule is composed of a number of macro decisions that partially specify actions of agents. This schedule is calculated according to a valid strategic decision shown in Table 5.1. A macro decision contains seven types of information:

1. A task type as a sub-goal which is scheduled to be done.
2. A set of agents who are allocated to this decision and are responsible for performing this decision.
3. A geographic object such as a road segment that presents the operational area of this decision.
4. A time that indicates when this decision starts to be executed.
5. A time that indicates when this decision ends.
6. A quantity of tasks which are estimated to be done during this decision.
7. A quantity of tasks with different types which are estimated to be revealed in the end of this decision at the same location.

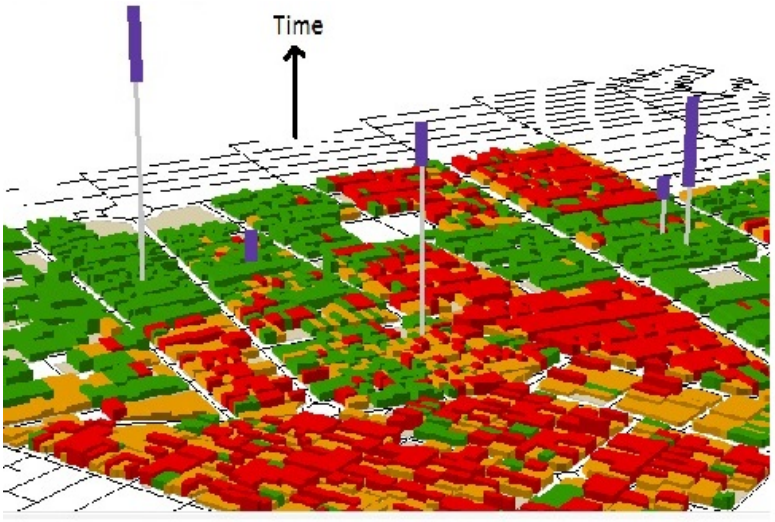
The adaption time is extracted from the schedule. It estimates a right time in which the current human decision will be required to be revised by the the I.C.. Also this time states that this schedule ends in this time, and a new schedule will be calculate based on the strategic decision adapted.

The overall time can be calculated if the scheduling algorithm and the planning algorithm are executed in sequence as Fig. 5.1 shows. Search algorithms can apply this method to calculate the total schedule with the estimated makespan in order to find a optimal solution for the scheduling problem.

Results showed that 1) the first human strategic decision was expired in time 579, 2) the total makespan is 16859, and 3) a sequence of twelve strategic decisions was made. For an example, Fig. 5.4 shows a geo-visualization of macro actions which are scheduled for an agent. An example of task assignment and action scheduling is presented in Fig. 5.5 for a anther data set in order to get better perception of results.

5.6.1 Discussion

This algorithm calculates a feasible, fast, and semi-optimal solution for the addressed problem. But, results showed that this algorithm does not guarantee optimality of



Agent Id	geoObject Id	Task Type	Start Time	Finish Time
ag2A1	se821	T0-Reconnaissance	0	95
ag2A1	se1157	T0-Reconnaissance	95	148
ag2A1	se1156	T0-Reconnaissance	148	344
ag2A1	se788	T0-Reconnaissance	344	428
ag2A1	se812	T0-Reconnaissance	428	579

FIGURE 5.4: a 3D map that geo-visualizes macro actions which are scheduled for a certain agent

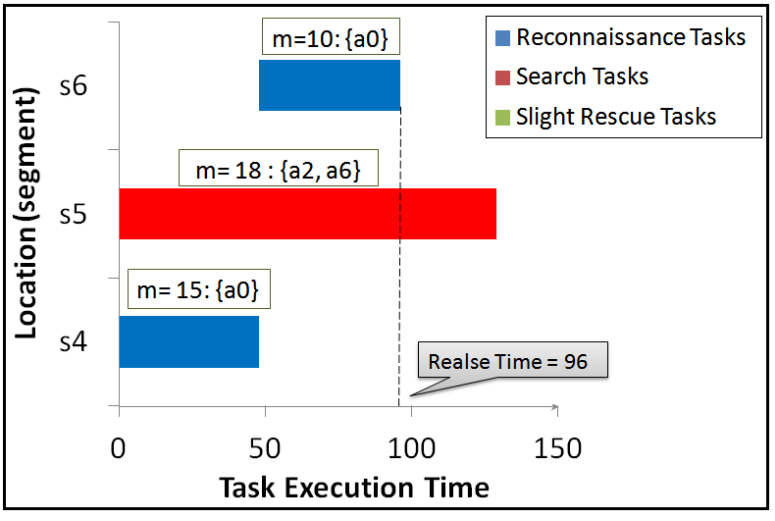


FIGURE 5.5: An example of assignment of macro tasks to agents for another data set.

assignments because tasks list showed that some agents sometimes become idle. In an efficient coordination, agents are busy and active. Minimizing the makespan is dependent

to the quality of five parameters: 1) human strategy, 2) strategic decision, 3) search algorithms, 4) scheduling algorithms, and 5) distribution coordination among agents.

The adaption time of a strategic decision is dependent to the structure of a scheduling algorithm and other factors. If several algorithms are applied on a same data, they will reach different results. We can not reason that the optimal assignment minimize the adaption time.

Tasks may include deadlines that play a major role in maximizing the global utility. Human can mange this issue by specifying a good high-level strategy.

This chapter proposed an algorithm for multi-agent scheduling. It seems a good idea to support the human by providing a set of algorithms for the I.C. and allow him to select a suitable algorithm for each thread. The system will apply a specific algorithm for a thread.

This chapter studied an I.C. while disaster emergency response may involve several incident commanders, and there are interdependencies among actions of teams. It seems important to study the distributed coordination among distributed incident commander and among their scheduling.

This chapter did not studied synchronized actions, the facilitating task dependency, and resource allocations in the proposed algorithm. These subjects may be important issues for some domains and teams.

A comprehensive approach for multiagent coordination includes a loop consisting of scheduling, execution, monitoring, adapting, and learning. Uncertain and dynamic environments disrupt scheduling, so it is necessary to execute decisions and continuously monitor execution and adapt scheduling to unexpected events. Learning algorithms can provide efficient tools to improve the efficiency and performance of this system.

5.7 Conclusion

This chapter presented a heuristic and greedy algorithm that appropriately assigns macro tasks to rational agents and schedules macro actions of agents according to the human strategic decisions. This algorithm was used to develop GICoordinator, which assistants the I.C. in coordinating a team of agents.

A schedule provides an I.C. a proper solution for the coordination problem in a team. Multi-agent scheduling partially specifies actions of rational agents and constrains them temporally and dynamically. the I.C. delegates agents to autonomously make their own

tactical decisions(planning and scheduling) or adapt their activities under these macro decisions.

This system executes this algorithm till the current strategic decision is valid. As a results, this algorithm calculates a time in which this strategic decision has to be adapted and revised. This result as useful information has two advantages: 1) It enables the I.C. to refine this strategic decision in a right time and 2) it is used in a search algorithm that estimates a makespan, calculate a complete solution, and find an optimal solution for multi-agent planning and scheduling.

Macro tasks enables the system to model and present distribution of tasks in different geographic scales. Spatial topology between spatial objects enables the system to extract new views of tasks, and these tasks can be used for task assignment. Up to quality of data, an I.C. can select different geographic layers such as zones or buildings and allocate tasks to them. So the I.C. can specify and contain actions in different spatial accuracies.

Information fusion algorithms extract new useful information from scheduling results. They provide an efficient situational awareness for an I.C. to percept states of global environment. They support human decisions by geospatial reasoning, geo-visualization, complex queries, etc. For example, they can answer these questions: What activities will done in a specific zone during a specific period? When a specific task will be done and by who? What is the task list of a specific agent?

Future works to improve the current contribution may include as follows:

1. In some cases, the tasks environment contains tasks that may require synchronous capabilities. It means that more that one agent have to coordinate their actions in order to provide the required capabilities for doing a specific task simultaneously. It is important for an ideal algorithm to form a proper coalition comprising of suitable agents and assign this task to this group.

2. Decentralized coordination of distributed schedules is a significant issue in where multiple teams are involved in performing tasks. It is possible that there are inter-dependencies between actions of teams. In order to maximize the joint objective, it is necessary to apply algorithms for coordinating distributed schedules which are made by each incident commanders.

References

[1] Chapin, Steve J. "Distributed and multiprocessor scheduling." ACM Computing Surveys (CSUR) 28, no. 1 (1996): 233-235.

[2] Chapman, Archie C., Rosa Anna Micillo, Ramachandra Kota, and Nicholas R. Jennings. "Decentralized dynamic task allocation using overlapping potential games." The Computer Journal 53, no. 9 (2010): 1462-1477.

[3] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[4] Davis, Ernest, and Jeffrey M. Jaffe. "Algorithms for scheduling tasks on unrelated processors." Journal of the ACM (JACM) 28, no. 4 (1981): 721-736.

[5] Fiedrich, Frank, Fritz Gehbauer, and U. Rickers. "Optimized resource allocation for emergency response after earthquake disasters." Safety Science 35, no. 1 (2000): 41-57.

[6] Gerkey, Brian P., and Maja J. Matari?. "A formal analysis and taxonomy of task allocation in multi-robot systems." The International Journal of Robotics Research 23, no. 9 (2004): 939-954.

[7] Horn, W. A. "Some simple scheduling algorithms." Naval Research Logistics Quarterly 21, no. 1 (1974): 177-185.

[8] Kwok, Yu-Kwong, and Ishfaq Ahmad. "Benchmarking and comparison of the task graph scheduling algorithms." Journal of Parallel and Distributed Computing 59, no. 3 (1999): 381-422.

[9] Kwok, Yu-Kwong, and Ishfaq Ahmad. "On multiprocessor task scheduling using efficient state space search approaches." Journal of Parallel and Distributed Computing 65, no. 12 (2005): 1515-1532.

[10] Lesser, Victor, Keith Decker, Thomas Wagner, Norman Carver, Alan Garvey, Bryan Horling, Daniel Neiman et al. "Evolution of the GPGP/TAEMS domain-independent coordination framework." Autonomous agents and multi-agent systems 9, no. 1-2 (2004): 87-143.

[11] Malone, Thomas W., and Kevin Crowston. "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) 26, no. 1 (1994): 87-119.

[12] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[13] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research, 2013. (under review)

[14] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." In Workshop on Robots and Sensors integration in future rescue INformation system, ROSIN'13. In conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), 2013.

[15] Opiyo, Elisha TO, Erick Ayienga, Katherine Getao, William Okello-Odongo, Bernard Manderick, and Ann Now. "Game theoretic multi-agent systems scheduler for parallel machines." (2008).

[16] Potts, Ci N. "Analysis of a linear programming heuristic for scheduling unrelated parallel machines." Discrete Applied Mathematics 10, no. 2 (1985): 155-164.

[17] Ramchurn, Sarvapali D., Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. "Decentralized coordination in robocup rescue." The Computer Journal 53, no. 9 (2010): 1447-1461.

[18] Scerri, Paul, Alessandro Farinelli, Steven Okamoto, and Milind Tambe. "Allocating tasks in extreme teams." In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 727-734. ACM, 2005.

[19] Vafaeinezhad, Ali Reza, Ali Asghar Alesheikh, Majid Hamrah, Reza Nourjou, and Rouzbeh Shad. "Using GIS to Develop an Efficient Spatio-temporal Task

Allocation Algorithm to Human Groups in an Entirely Dynamic Environment Case Study: Earthquake Rescue Teams." In Computational Science and Its Applications–ICCSA 2009, pp. 66-78. Springer Berlin Heidelberg, 2009.

[20] Wen, Yun, Hua Xu, and Jiadong Yang. "A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system." Information Sciences 181, no. 3 (2011): 567-581.

Chapter 6

Search-based Algorithm for Optimal Strategic Decision Making in Spatial Multi-agent Planning

6.1 Abstract

Problem: This chapter addresses an optimization problem in spatial multi-agent planning. In order to coordinate a team of spatial agents (robots or field units) to achieve a joint objective in the crisis response, the responsibility of an incident commander is to specify a strategy composed of threads (a set of prioritized sub-goals), make a SD (Strategic Decision) by appropriate assignment of agents to threads, constrain actions of agents to allocated threads, continuously monitor the state of the world, and timely adapt this SD to the new situation and re-make a new one. The research question of this chapter is how to make the best strategic decision by an GIS-based intelligent assistant system that it can guarantee to achieve an objective from the initial state in a minimum overall time (cost)?

Objective: This chapter intends to present a state space search algorithm that can select the best choice as the semi-optimal strategic decision from a set of previously calculated alternatives in a definite time. In addition, it estimates a minimum overall time which the team needs to achieve the objective.

Method: Our methodology is a A* search algorithm. It first generates a node per each alternative, then it executes two sub-algorithms sequentially to : 1) assign tasks to

agents according to the SD of the node **until** this SD requires to be adapted and 2) calculate the h, which is an estimation of total time required by team to reach the goal, and then f. Finally, generated nodes are added to the search space. Then a node with minimum f is selected from the state space. If the selected node is not the goal state, the algorithm calculates a new set of feasible alternatives for revising/adapting the SD associated with this node; and this procedure is repeated. For the goal node, the state space is explored to identify an initial alternative from which this gaol node is generated. This alternative is the best SD.

Results: For a simulated problem, three results were achieved: 1) an alternative as the best choice which was selected from a set of available alternative, 2) a time which indicates the minimum overall time required by agents to reach the goal, and 3) an optimal plan which was a sequence of the selected alternative and calculated strategic decisions.

New Findings: A selected alternative is the best SD that the IC can make for the strategic decision making problem. Results support human decisions and assists him in action planning for spatial agents. This algorithm can be used to develop intelligent GIS that collaborate with human for making optimal strategic/macro decisions in order to coordinate actions of spatial agents in uncertain, dynamic, geographic environment such as quick response management.

6.2 Introduction

Multi-agent planning is a significant issue in multi-agent systems for action coordination [7]. Advanced algorithms is of importance for development of intelligent GIS. Strategic decision-making provides a centralized method for the IC (incident commander) to coordinate and control actions of a team of spatially distributed agents in order to achieve a global objective in the domain of disaster crisis response [9]. The responsibility of the IC includes: 1) specify a strategy by decomposing the whole planning problem into a set of prioritized sub-goals or subproblems which are called threads, 2) make a SD (strategic decision) by appropriate assignment of agents to threads, 3) disseminate information of this SD among agents, 4) continuously monitor the world's state and task execution, and 5) timely adapt/revise this SD to the new situation.

A SD states which agents are assigned to which threads [1]. Because a thread is a subproblem, a SD does not fully specify agents' actions but it temporally constrains agents to threads and delegate these agents to explicitly make their own decisions about their actions for doing tasks according to this SD. In another word, a SD which is made by the

IC as a central planner should be executed by distributed agents. In a state of the world, the IC is faced with a number of feasible alternatives for making a SD, that selection of each one may have a significant effect on the coordination problem. A sequence of strategic decisions enables a team to get from the initial state to the goal state.

In real-time, making an optimal SD is an important and essential issue for a team. This requires the IC to select an optimal SD from a set of feasible alternatives which is calculated for that time. A sequence of strategic decisions generates a plan, and an optimal plan guarantees agents to achieve a joint and high-level objective in a minimum time or maximize a global payoff through a sequence of optimal strategic decisions. Because of uncertain and incomplete information, in real-time, the IC makes (or re-makes or adapts) an optimal SD offline, and then agents execute this decision online, and this procedure is repeated till the goal state is reached. The research question of this chapter is how to select/find the best choice an optimal SD from a set of feasible alternatives which is calculated for making a SD for a definite time?

Optimal strategic decision-making is an important optimization problem in spatial multi-agent planning. Our motivation is to address this problem because of the following facts:

- Disaster response management is a complex environment that contains incomplete, dynamic, geographic, and uncertain information, and efficient coordination is a significant issue for this domain [3]. Because of this complexity, it is difficult and hard for the IC to reason which alternative can be the best SD.
- An IC is continuously faced with the problem of re-making/revising a SD to new crisis situations in real-time. This is difficult for human to efficiently do this responsibility under time pressure during emergency response operations.
- Strategic decisions v.s. the tactical decisions, which are made by agents, define the macro behavior of a team. It is essential for the IC to make the best one in each time.
- An IC needs to approximatively estimate a minimum time in which the high-level objective will be reached by accomplishing a sequence of optimal strategic decisions. It enables the IC to evaluate the quality of a defined strategy in order to adjust this strategy or respecify a better strategy.
- Under uncertainty, first, an optimal SD is made by the IC, then it is executed by agents. The IC, who has a timely situational awareness, is required to re-make a new SD according to the updated state of the world. This procedure is repeated till the objective is reached by the team. It is a hard and challenging task for the human.

As a result, it is important and necessary to develop an intelligent assistant system that assists human in the optimal SD making problem in complex situations such as disaster crisis management. The role of this system is to select the best choice from a set of alternative scenarios, which is previously calculated and available for this system for the SD making problem. Over time, this system should autonomously re-make (or revise) new optimal strategic decisions. In a state of the world, this system is required to calculate and present two kinds of information: 1) one SD which is the first SD of the optimally calculated plan and 2) an approximate estimation of total time in which the objective will be reached by the team via this optimal plan.

This requires a proper approach especially in problem-solving methods. There is much literature in state space search algorithms, planning, scheduling, decision making, geospatial analysis, task assignment, GIS, spatial decision support systems, optimization, and multi-agent coordination. Unfortunately these works do not thoroughly address the problem stated by this chapter or can not provide a proper solution for this problem. One of the most widely-used methods for problem-solving in artificial intelligence is state space search [13]. A state-space search problem is defined by a set of states, a set of actions (or operators) that map states to successor states, a start state, and a set of goal states. A* finds a solution that takes the form of a sequence of actions leading in a path from the start state to a goal state. The objective is to find a sequence of actions that transforms the start state into a goal state, and also optimizes some measure of the quality of the solution. One well-known heuristic search algorithm for state-space search problems is A*. Because of the complexity of SD making, we need to integrate some techniques in order to develop a proper algorithm for solving this problem.

The contribution of this chapter is a A* search algorithm for optimal SD making. Two main issues are of importance to apply this algorithm in multi-agent planning: 1) select the best choice as the optimal SD from a set of alternatives and 2) estimate a minimum time by which the the team will reach the high-level objective. In fact, this algorithm calculates an optimal plan whose first SD is one of these alternatives. An intelligent assistant system that is equipped with this algorithm can support/assist human (the IC) in SD making during disaster emergency response operations for coordination of agents.

This chapter is organized as follows. To get better insight, section 2 provides our motivation by introducing a simulated urban search & rescue scenario in which the IC of a team of four agents is faced with the coordination problem of agents. Section 3 reviews some related works. The approach and the proposed algorithm are presented in section 4. Section 5 is dedicated to achieved results. Finally, we have a conclusion.

6.3 Motivation

Imagine that an earthquake disaster has occurred in an urban area. Fig. 6.1 shows a GIS map of the initial state of a disaster-affected area to the IC. This map visualizes spatial distribution of LoTeM tasks (Location-based Temporal Macro tasks) in five operational zones (road segments) and also the initial location of a disaster response team. Urban search and rescue (USAR) as a major function of disaster emergency response operations aims to rescue the greatest number of people who are trapped under the debris of damaged buildings in the shortest amount of time. To save one victim, who is located at a damaged building in a certain spatial location, a sequence of three dependent tasks should be accomplished, and each task requires a set of definite capabilities and a considerable amount of time. Table 6.1 lists three task types and capabilities requirements in the domain of USAR. It is obvious that USAR is required to save many people, who are affected e.g. by earthquake disaster, by accomplishing a number of tasks. Real tasks which are spatially distributed in the area are instances of these three task types. We ignore the deadline for tasks in this chapter.

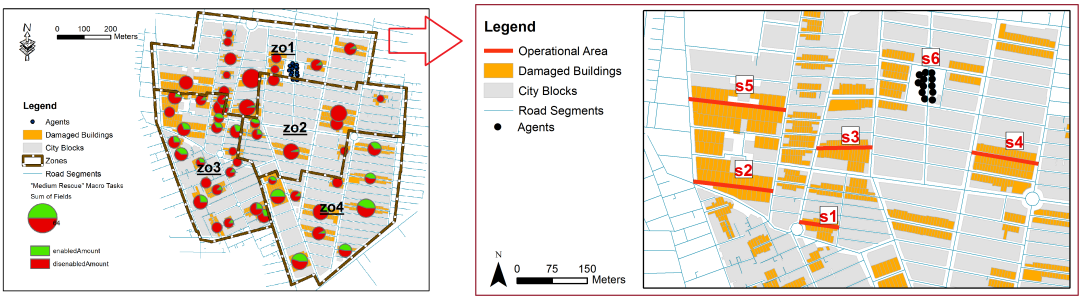


FIGURE 6.1: A GIS map of the initial state of the world to show a provide the timely situational awareness for the IC.

TABLE 6.1: Three task types of the problem domain.

Task-Type	**Δ *t* (minute)**	**Capability Requirements** ***C0***	***C1***	***C2***
T0	5	1	0	0
T1	20	0	1	0
T2	60	0	0	1

Capabilities Description: **Task Types Description**:
C0: Reconnaissance **T0**: Reconnaissance
C1: Search **T1**: Search
C2: Light Rescue **T2**: Light Rescue

A team of four agents (field units) has been assigned to these five operational areas which are presented by five road segments displayed in Fig. 6.1. Actions which a definite agent can do are presented by capabilities that this agent possesses. An agent can have different capabilities which are required by tasks and consequently this agent may have

different actions. In each time, this agent can select an action from its own actions and execute it with a definite speed. Table 6.2 lists characteristics of these agents. It was assumed the agents are *free* (or idle) and are located in the incident command post.

TABLE 6.2: Capabilities/abilities of four agents.

Field Unit ID	**action speed**	**Number of Capabilities** *C0*	*C1*	*C2*
a0	2	1	0	0
a2	1	1	0	0
	2	0	1	0
a6	1	0	1	0
	2	0	0	1
a7	1	0	1	0
	2	0	0	1

Shortest distances among six road segments are calculated by GIS and are listed in Table 6.3. To do spatially distributed tasks, agents need to move from one location to another through the road network with the moving speed e.g. equal to 20 (meter per minute). To make the problem simple, this table does not change over time. These data are used by the IC for decision making. This table and related information are provided and are updated by relevant teams or organizations whose activities are to clear road blockages.

TABLE 6.3: The shortest distances (given in meter) among six road segments displayed in Fig. 6.1.

	s2	**s1**	**s3**	**s4**	**s5**	**s6**
s2	0	225	447	764	364	625
s1	225	0	370	687	418	548
s3	447	370	0	343	452	221
s4	764	687	343	0	618	224
s5	364	418	452	618	0	476
s6	625	548	221	224	476	0

A set of LoTeM tasks are associated to each road segment. Simply, a LoTeM task is an aggregative task of all tasks that are with a same task type and are spatially located within a geographic area. Table 6.4 shows that twelve LoTeM tasks are located to the five operational areas (segments) at the initial time 0. For example, the 10th LoTeM task states that the proximity of the road segment "s5", five light rescue tasks are estimated to be revealed/enabled at future and six tasks are available and ready to be done by agents. There is the "enabling" dependency between the 9th LoTem task and 10th LoTeM task. It states and estimates that if the whole 9th LoTeM task (2 not yet enabled amount plus 5 enabled amount) is completely accomplished, these five light rescue tasks will be revealed in the same location. Agents can do only a subset of tasks that has been enabled before. Finally, all these information is presented for the IC in order to provide a timely situational awareness of the world's state. Information of these tasks forms the big picture that the IC observes from the task environment.

TABLE 6.4: Information of a set of LoTeM tasks associated with the five road segments shown in Fig. 6.1.

No.	Location (Road S.)	Task-Type	Not Yet Enabled Amount	Enabled Amount
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	15
6	s3	T1	8	0
7	s3	T2	2	8
8	s5	T0	0	10
9	s5	T1	2	5
10	s5	T2	5	6
11	s4	T1	0	18
12	s4	T2	10	2

6.3.1 Spatial Multi-agent Planning by the IC

First step in coordination of agents by the IC is to specify a strategy. An joint (or high-level) objective is a desirable state of the world that a team aims to achieve with regard to some criteria. The objective which the IC selects is to rescue all victims distributed in the five operational areas. It means that all light rescue tasks (enabled tasks plus not yet enabled tasks), which are shown in Table 6.4, should be carried out. As Table 6.5 presents, the IC specifies a strategy which is composed of three threads in order to achieve the defined goal. Thread 1 states that the first and the highest priority for the team is to do reconnaissance and search operations at three geographic areas {s3, s4, s5}, and any *appropriate* and *available* subset of four agents {a0, a2, a6, a7} can be assigned to this thread in order to accomplish tasks that this thread defines/contains. The human strategy has defined the agent a2 for three threads; it means that this agent should be allocated to one thread via a SD, and this agent will be available for the next thread whenever the higher thread releases this agent. To specify a strategy, the IC takes into account two important facts: 1) agents availability for threads and 2) tasks dependencies among threads. These factors make threads interdependent completely or partially. For an example, the search LoTeM task of the first thread enables/reveals the light rescue LoTeM task of the second thread. We assume that the IC has sent all these free agents to the thread 1.

TABLE 6.5: The human strategy with three threads for coordination of agents' actions

Thread Id	a sub-Location	a sub-Goal (Task Type)	a sub-Team
1	s3, s4, s5	T0, T1	a0, a2, a6, a7
2	s3, s4, s5	T2	a2, a6, a7
3	s1, s2	T0, T1, T2	a0, a2, a6, a7

The second step is to execute the human strategy by appropriate assignment agents to threads that it will result in a SD. The IC is required to make a SD in two steps: 1) calculate a set of feasible alternative and 2) select a choice from these alternatives as the final SD. According to the described situations in the initial state, Table 6.6 shows a set of alternatives which has been calculated [1]. For example, if the last alternative is selected as the final SD, this decision will constrain all four agents to the thread 1; and it means that four agents are allowed to do any reconnaissance or search tasks which are spatially distributed in three geographic areas {s3, s4, s5}. This decision does not defines who should do what task, where, and when explicitly and also it does not defines that when the defined objective will be achieved.

TABLE 6.6: Ten alternatives which were calculated for making a strategic decision in the initial time 0 [1].

Choice No.	**Thread Assignment 1**	**Thread Assignment 2**	**Thread Assignment 3**
1	a2	a7	a0, a6
2	a2	a6, a7	a0
3	a0, a7	a6	a2
4	a2, a7	a6	a0
5	a0, a2	a7	a6
6	a0, a2	a6, a7	
7	a0, a2, a6		a2
8	a2, a6, a7		a0
9	a0, a2, a7	a6	
10	a0, a2, a6, a7		

6.3.2 The Optimization Problem: The Optimal SD Making Problem

To do the second step, the IC is faced with some crucial questions:

- Which alternative may be the best/optimal SD in this time?
- What is the minimum overall cost (time) by which my team will probably reach the goal state (the defined objective) according to this alternative?

The objective of the optimization problem of SD Making is to select the best alternative as the optimal SD. It will enable a team to reach the goal state in a minimum total time (cost) of task execution. The objective function will be as follows, and the best alternative will be the first SD of the sequence of strategic decisions.

$$\sum_{i=1}^{n} SD = \underset{p}{\operatorname{argmin}}\, \Delta t$$
$$\Delta t : total\_time\_of\_tasks\_execution \tag{6.1}$$

.

A SD is **not** the problem of assignment of tasks to agents that results in a task (or action) schedule. We should consider that a SD is not an action that changes the state of the task environment. It defines the domain of actions of a subset of agents by assigning this subteam to a specific thread. The scheduling problem should be solved according to a made SD that is valid for that time.

6.4 Literature Review

Strategic planning is a coordination approach for managing tasks relationships by objectives (goal selection and goal decomposition) and grouping people into units [7]. In the organization theory, strategic management consists of four basic elements: environmental scanning, strategy formulation, strategy implementation, and evaluation and control in order to achieve organizational objectives. [4]. This approach is used by ICS (incident command systems) [2]. Unfortunately, these works do not state how to make optimal decisions.

STaC is the first approach that proposed the human strategy guidance in multiagent coordination [6]. It is used for automated adaption of strategic decision by assigns a subset of available agents with a minimum capabilities to threads. It is a considerable inefficiency in this approach because the optimal decision may be to assign a subset of available agents to a specific thread that requires maximum capabilities.

Dialogue-Assisted Visual Environment for Geoinformation was proposed as an approach for designing natural, multimodal, multiuser dialogue-enabled interfaces to geographic information systems that make use of large-screen displays and integrated speech–gesture interaction [5]. However it was used in the emergency response centre, unfortunately it does not address the decision making problem for the field unites.

Different versions of search algorithms have been proposed for solving different types of problems. Two examples include routing with stopover areas [14] and finding shortest paths on real road networks [15]. It is important to modify a search algorithm in order to solve a specific problem with definite characteristics.

6.5 Methodology

Algorithm 1 presents our approach that is proposed for the optimal SD making problem. It is a A* search algorithm for selecting a best choice from a set of alternatives in the optimal SD making problem. It includes some sub-algorithms. A sub-algorithm

first generates a node for each alternative for the initial time and adds this node, which contains a different SD, to the state space. The state space is a search three that is expanded by this sub-algorithm and contains the generated nodes. For each newly generated node, two types of sub-algorithms are run sequentially: 1) a heuristic algorithm that assigns LoTeM tasks to agents according to the SD associated with this node **until** this SD requires to be revised/adapted by releasing a subset of agents from a subset of threads and 2) a heuristic algorithm that calculates the h, which an estimation of amount of time that is required by agents to reach the goal state. Then the algorithm searches the state space in order to select a node with the minimum f. Finally, if the selected node is not the goal state, an algorithm will be used to calculate a new set of feasible alternatives for revising/adapting the SD associated with this node; and this procedure is repeated. If the algorithm reaches the goal node, a sub-algorithm explores the state space in order to identify an initial alternative as the the root node from which the gaol node is generated. This alternative indicates to the optimal SD.

Data: A :a set of alternatives which is used to optimize the SD making problem.
Data: $n0$:the initial node, which is an instance of the "stateNode" class.
Result: SD :the best alternative as the optimal SD.
Result: t :a minimum overall time (cost).
$StateSpcae \longleftarrow \emptyset$;
while $true$ **do**
for $a \in A$ **do**
$n \leftarrow Generate\_aNew\_Node(a, n0)$;
$Assign\_LoTeMTasks\_toAgents(n)$;
$Calculate\_h(n)$;
$StateSpcae \longleftarrow StateSpcae \cup \{n\}$;
end
$n \leftarrow Select\_a\_Node(StateSpcae)$;
if $n.g2 = null$ **then**
$SD \leftarrow null$;
return [null, null] ;
end
if $Is\_the\_goalNode(n) = true$ **then**
$n2 \leftarrow Find\_theRoot\_Node(n, StateSpcae)$;
$SD \leftarrow n2.ThreadAssignments\_node$;
$t \leftarrow n.g2$;
return [SD, t];
end
$n0 \leftarrow n$;
$A \longleftarrow Calculate\_aNewSet\_Alternatives(n0)$;
end

Algorithm 4: The A* search algorithm for the optimal SD making problem in coordination of spatial agents.

We require a data model to formulate and model the problem. The SAP data model [9] was used by this chapter to develop this algorithm. Fig. 6.2 shows a part of the SAP data model which includes the strategy, state node (or node simply), strategic decision or alternative, thread assignment, and thread classes.

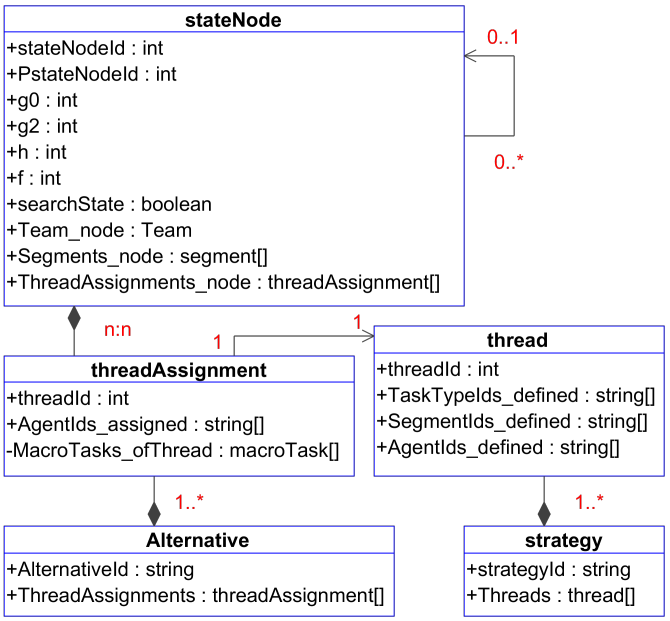


FIGURE 6.2: A part of the SAP data model [9].

The proposed algorithm integrates a number of sub-algorithms to solve the main problem. The following subsections are dedicated to methods/algorithms used in the main algorithm.

6.5.1 Sub-algorithm: Generate a New Node

The objective of this algorithm is to generate a new node per each available alternative using the initial node. A newly generated node contains a different SD, but it does not change the state of the task environment. T attributes of the "stateNode" class are described as follows:

- **g0** is a time in which this node is generated. The g2 of the parent node is assigned to this attribute.
- **g2** is a time that this node ends (or finishes). This parameter indicates when the SD should be adapted.
- **f** is an estimation of total time from the root node to the goal node.
- **h** is an estimation of cost (amount of time) that is required by the agents to reach the goal state from the current node.
- **ThreadAssignments_node** is comprises a set of instances of the "threadAssignment" class. It presents/models a SD that is associated with this node. It is valid during the lifetime of this node.

- **Segments_node** encodes LoTeM tasks.

If we execute this algorithm on the problem stated in Section 2, 10 new nodes will be generated. The "g0" property of all these nodes is set into 0 that points to the initial time.

6.5.2 Sub-algorithm: Assign LoTeM Tasks to Agents in a Node

Algorithm 2 presents the sub-algorithm. It comprises several sub-methods. It is a heuristic algorithm for assignment of LoTeM tasks to agents in a node according to the node's SD. It intends to calculate (or estimate) a right time at which the node' SD should be adapted by releasing a subset of agents from the a subset of threads. The "Segments_node" and "g2" properties of the node will updated by this algorithm. This algorithm has been discussed in detail by [11].

If this algorithm is executed on the 6th node, which contains the 6th alternative of Table 6.6, we will achieve an action schedule presented in Fig. 6.3. According to this result, the agent "a0" should be released from its thread at time 96 because the thread 1 does not need to keep this agent anymore. The g2 property of this node should be updated to 96. Table 6.7 shows the updated state of the LoTeM tasks (the "Segments_node" property) at the time 96.

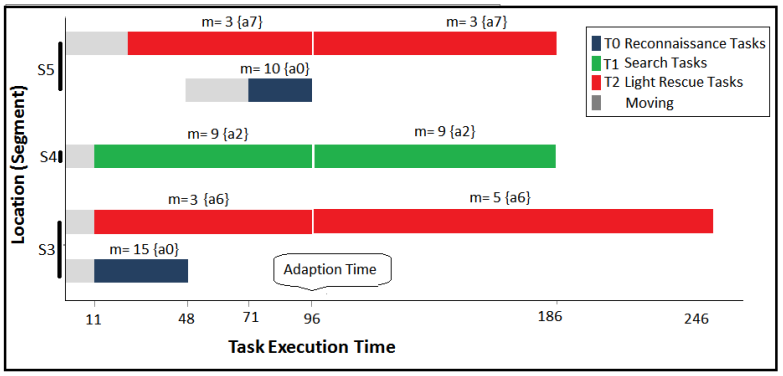


FIGURE 6.3: The task/action scheduling in the 6th Node.

6.5.3 Sub-algorithm: Calculate the "h" of Node

A heuristic algorithm is used to calculate the "h" property of a node. Fig. 6.4 shows an algorithm that calculates a total dependent duration which is associated to a definite LoTeM task using two types of information: 1) amount of the enabled tasks and 2) amount of dependent tasks that will be revealed by accomplishing this LoTeM task. It is considerable that if a set of agents is assigned to a definite LoTeM task, as Fig. 6.3

```
g2 ← g0;
L ⟵ Create_emptyset_of_legalAssignment();
while true do
Select_Efficient_Agents();
T ⟵ Select_Active_MacroTasks();
A ⟵ Select_Idle_Agents(g2);
L ⟵ L ∪ A;
if Identify_theRelease_Time() = true then
return;
end
while true do
if |L| = 0 or |T| = 0 then
break;
end
T2 ⟵ Nominate_MacroTasks();
U ⟵ Calculate_Utilities();
u ⟵ Find_theHighest_Utilities();
if u.benefitRation >= 10 then
Assign_Agents_toMacroTasks();
continue;
end
else
T ⟵ T − T2;
continue;
end
end
if |L| = 0 and |T| > 0 then
g2 ← Calculate_earliestFinishTime();
continue;
end
else if |L| > 0 and |T| = 0 then
if Identify_theRelease_Time() = true then
return;
end
else
g2 ← Update_ProblemState();
if g2 = null then
return;
end
else
continue;
end
end
end
end
```

Algorithm 5: The heuristic algorithm for dynamic assignment of LoTeM tasks to agents within a node [11].

shows, the finish time of this task will be used in calculation of this parameter. The "h" variable of a node is the aggregation of all total dependent durations of LoTeM tasks. Consequently, the "f" property of this node will be the aggregation of the "h" and "g2" properties. Table 6.8 will be achieved if we execute this algorithm for the LoTeM tasks shown in Table 6.7.

TABLE 6.7: The updated state of the LoTeM tasks in the 6th node at time 96.

No.	Location (Road S.)	Task-Type	Not Yet Enabled Amount	Enabled Amount
1	s1	T0	0	25
2	s1	T2	0	4
3	s2	T1	0	5
4	s2	T2	10	5
5	s3	T0	0	**0**
6	s3	T1	**0**	**8**
7	s3	T2	2	**5**
8	s5	T0	0	**0**
9	s5	T1	**0**	**7**
10	s5	T2	5	**3**
11	s4	T1	0	**9**
12	s4	T2	**5**	**7**

$$
\begin{aligned}
N &= |LoTeMTasks| \\
h &= \alpha * (\sum_{i=1}^{N} DTT) \\
f &= g2 + h
\end{aligned}
\tag{6.2}
$$

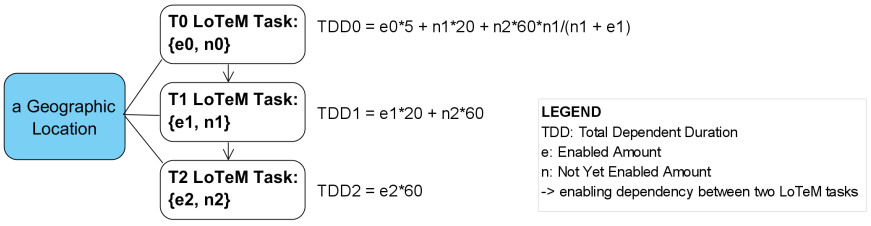


FIGURE 6.4: A heuristic algorithm for calculation of the total dependent duration of LoTeM tasks regarding the problem domain.

TABLE 6.8: The total dependent duration of LoTeM tasks shown in Table 6.7.

No.	Total Dependent Duration
1	125
2	240
3	700
4	300
5	0
6	280
7	150
8	0
9	440
10	90
11	390
12	420

6.5.4 Expand the State Space

Now, 10 new nodes were generated and were completely calculated. These nodes will be added to the state space in order to expand the search tree. Fig. 6.5 presents this search three with the 10 nodes.

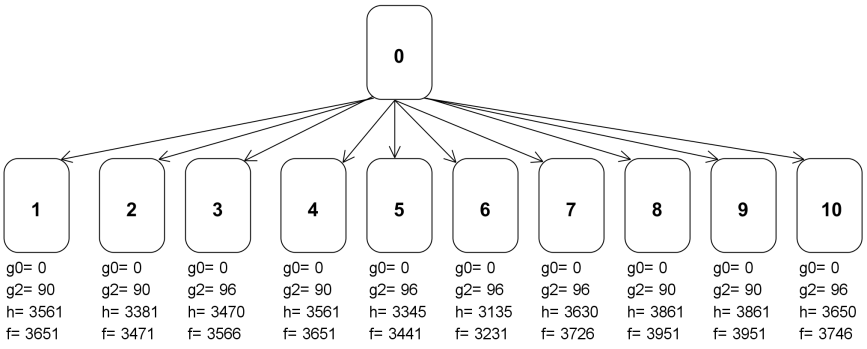


FIGURE 6.5: The search tree including 10 newly generated nodes.

6.5.5 Sub-algorithm: Select a Node

The purpose of algorithm is to select a node from the state space. There may be different methods to select a node according to some criteria. A selected node will be used for two purposes: 1) to select as the goal state or 2) to expand the search tree. Our method in this chapter is to select a node with the the smallest f from the available nodes. This method is used in A* search algorithms. According to this method, the node 6 will be selected from of the state space shown in Fig. 6.5.

6.5.6 Sub-algorithm: Find the Root Node

This algorithm will be executed if the selected node is recognized as the goal node. The objective of this algorithm is to identify a root node that reaches this goal node through a sequence of nodes within the state space. Because each node keeps the id of its parent in the search tree, it is easy to tracks nodes from end to beginning.

In addition, the "g2" property of this node presents the minimum overall time (cost) that the team can reach the objective from the initial state. The sequence of nodes generates a plan.

6.5.7 Sub-algorithm: Calculate a New Set of Alternatives

Algorithm 3 is used to calculate a set of new alternative for adapting/revising the SD of the selected node at time "g2". The "g2" property indicates a right time that one thread or a subset of threads should release a subset of their agents and send them into next threads. We are faced with the problem of re-assignment of these agents to threads. A thread, which receives new agents, can select a sufficient subset of these agents and sends unwanted ones into the lower thread, but maybe there is a set of different subsets as available options for this thread. Therefore, a set of feasible alternative scenarios may be existent that present different kinds of distribution of agents among threads. An alternative is a candidate for the final SD. This algorithm has been discussed in detail by [1].

Data: n :an entity of the "stateNode" class of the data model.
Data: S :an entity of the "strategy" class.
Data: D :the problem domain.
Data: p :type of selection method.
Result: N :a set of entities of the "stateNode" class that present feasible alternatives.
for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
$t \leftarrow S.Threads[i]$;
$ta \leftarrow n.ThreadAssignments\_node[i]$;
$ta.MacroTasks\_ofThread \longleftarrow f\_Calculate\_MacroTasks(n.Segments\_node, t, D)$;
end
$Na \longleftarrow \emptyset$;
$Nb \longleftarrow \emptyset$;
$Nb \longleftarrow Nb \cup \{n\}$;
for $i \leftarrow 1$ **to** $|S.Threads|$ **do**
$t \leftarrow S.Threads[i]$;
for $nb \in Nb$ **do**
$ta \leftarrow nb.ThreadAssignments\_node[i]$;
$A1 \longleftarrow f\_Identify\_Agents\_ResidentIn(ta)$;
$A2 \longleftarrow f\_Identify\_Agents\_ReceivedBy(ta)$;
for $m0 \in ta.macroTask\_ofThread$ **do**
$tm0 \leftarrow m0.TemporalMacroTasks.Last()$;
$tm0.LegalAssignments \longleftarrow f\_Select\_Efficient\_Agents(m0, t, A1, A2)$;
end
$M \longleftarrow ta.macroTask\_ofThread$;
$C1 \longleftarrow f\_Form\_EfficientCoalitions(M, A1, A2)$;
$C2 \longleftarrow f\_Purify\_Coalitions(C1, M)$;
$C3 \longleftarrow f\_Select\_Coalitions(C2, p)$;
for $j \leftarrow 1$ **to** $|C3|$ **do**
$Na \longleftarrow Na \cup \{f\_Generate\_NewNode(C3[j], nb)\}$;
end
end
$Nb \longleftarrow Na$;
$Na \longleftarrow \emptyset$;
end
$N \longleftarrow Nb$;

Algorithm 6: The automated algorithm for calculation of a set of feasible alternatives in making a SD [1].

As Fig. 6.3 shows, the SD of 6th Node should release the agent "a0" from the 1th thread

at time 96. The Fig. 6.6 presents a set of new feasible alternatives that is calculated by running this algorithm for this node. Newly calculated alternatives will be used to expand the state space. Fig. 6.7 shows the state space which is expanded by the new alternative.

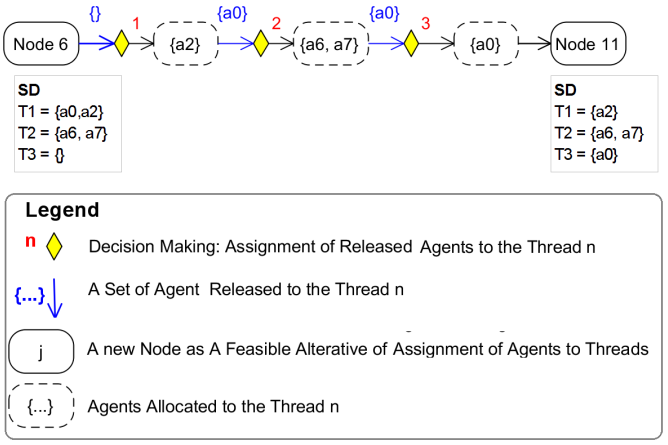


FIGURE 6.6: Calculation of new alternatives by adaption of the SD of 6th node at time 96 [1].

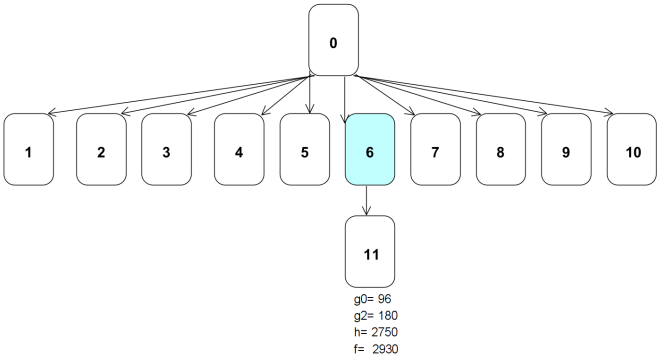


FIGURE 6.7: The search tree expanded from the node 6.

6.6 Implementation

The proposed algorithm was implemented in GICoordinator. GICoordinator is a GIS-based intelligent assistant system that assists the IC and supports human decisions in coordination of a team of field units in crisis response management in especial in urban search & rescue operations [10]. This spatial intelligent system supports the IC with intelligent algorithms for action planning and task scheduling for centralized coordination of a team in a dynamic and spatial environment [11**?**]. Also, it applies a spatial database for geographic and location-based information management and uses GIS functions to support development of these spatial intelligent algorithms [9, 10]. The C#.Net programming language has been used to implement the core of the GICoordinator. An

TABLE 6.9: The best alternative which was selected as the optimal SD in the SD making problem, which was stated in Section 2.

The Selected Alternative	**Minimum Total Time (in minutes)**	**Computation Time (in milliseconds)**	**Number of Generated Nodes**	**Search Method**
6	894	101	17	A* with $\alpha = 1.0$
6	894	261	74	Breadth-First

TABLE 6.10: An optimal plan which is composed of a sequence of calculated strategic decisions to reach the goal state from the initial state.

No. of the SD	**Agents Assigned to Thread 1**	**Agents Assigned to Thread 2**	**Agents Assigned to Thread 3**	**Start Time (Assignment Time)**
1 (the Alternative 6)	a0, a2	a6, a7		0
2	a2	a6, a7	a0	96
3	a2	a6, a7		180
4		a6, a7	a2	393
5		a6, a7		463
6			a6, a7	576
7				894

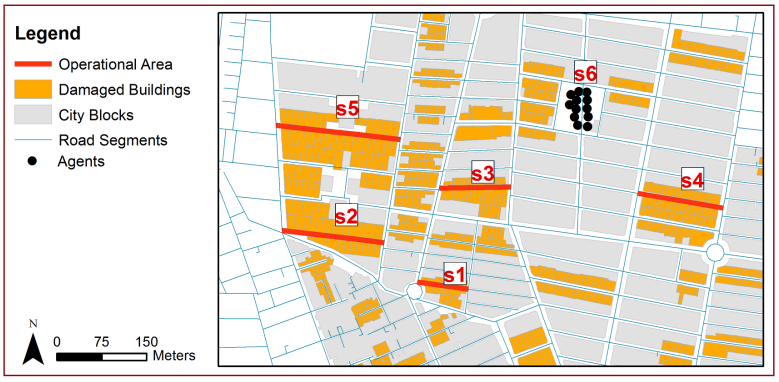
IC is equipped with a computer that runs an instance of GICoordinator. A simulator that runs multiple instances of the GICoordinator was developed to be used for evaluation distributed algorithms for decentralized coordination [12]. A simple version of this agent has been developed for the field units [8].

This algorithm was executed for the scenario which was stated in Section 2 in order to get a better understands. In addition, we tested the presented algorithm on various human strategies and different number of agents in order to calculate the run time (computation time) and number of generated nodes. Results are shown in the next section.

6.7 Results

The results which are shown by Table 6.9 and 6.10 have been calculated by executing the Algorithm 1 for the problem stated in Section 2. They state two main sub-results: 1) the alternative 6 of Table 6.6 is the best choice among ten available alternative scenarios for optimal SD making at initial time 0 and 2) the high-level objective defined by the IC will be reached by agents at the time 894 (in minutes) as the minimum overall time. As a result, the alternative 6 is the best one that should be selected by the IC for multi-agent planning. Fig. 6.8 presents an integration of the optimal SD with the GIS map for providing a better situational awareness for the IC.

In addition, two minor results are considerable: 1) according to the node 6, it is estimated that the IC will be required to adapt the SD at time 98 and 2) this result enables us to extract an optimal plan, which comprises a sequence of strategic decisions, from the state space. This results can be used by the IC to evaluate the efficiency of a strategy.



Thread Id	a sub-Location	a sub-TaskTypes	Assigned Agents	Start Time
1	s3, s4, s5	T0, T1	a0, a2	0
2	s3, s4, s5	T2	a6, a7	0
3	s1, s2	T0, T1, T2		0

FIGURE 6.8: Visualization of the optimal SD which was made for coordination of spatial agents in a geographical area.

6.7.1 Evaluation

To evaluate the proposed algorithm, we executed it on the same LoTeM tasks with three types of strategies, three different sizes of a team, three different search methods. Three kinds of strategies were defined: 1) the strategy of type I includes three threads that are independent in agents, 2) the strategy of type II includes three semi independent threads, and 3) the strategy of type III which is a complex strategy is composed of three threads that all agents are defined for all threads. In addition, three search methods, which were used by the main algorithm to search the search space in order to select a node, include: 1) the A* search method with $\alpha = 1.0$, 2) the A* search method with $\alpha = 0.3$, and 3) the Breadth-First algorithm, which is a kind of the A* search method with $\alpha = 0.0$.

Table 6.11 shows results of this evaluation. The result achieved for each scenario includes: 1) the computation time (run time), 2) number of generated nodes, and 3) the minimum overall time of task execution.

The results show that the however the breadth-first search method can guaranty the optimality of the problem, but it is should generate a big state space including a huge number of nodes in order to find the goal node. Consequently it is impossible to use for a complex problem. The A* search method finds a feasible, semi optimal solution in a short time, but for the a complex problem, we should apply a suitable variable α.

TABLE 6.11: Evaluation of the proposed algorithm.

Team Size	**Strategy Type**	**Minimum Total Time (minutes)**	**Computation Time (millisec.)**	**Number of Generated Nodes**	**Search Method**
4	I	1074	5	94	A* with $\alpha = 1.0$
4	I	1074	5	93	A* with $\alpha = 0.3$
4	I	1074	5	91	Breadth-First
4	II	1104	7	95	A* with $\alpha = 1.0$
4	II	1104	7	97	A* with $\alpha = 0.3$
4	II	1074	10	100	Breadth-First
4	III	931	9	102	A* with $\alpha = 1.0$
4	III	931	9	97	A* with$\alpha = 0.3$
4	III	931	22	141	Breadth-First
8	I	725	12	99	A* with $\alpha = 1.0$
8	I	530	12	107	A* with $\alpha = 0.3$
8	I	530	25	138	Breadth-First
8	II	541	12	105	A* with $\alpha = 1.0$
8	II	541	12	103	A* with $\alpha = 0.3$
8	II	540	32	167	Breadth-First
8	III	1082	9	109	A* with $\alpha = 1.0$
8	III	729	15	121	A* with $\alpha = 0.3$
8	III	558	38	168	Breadth-First
12	I	714	12	95	A* with $\alpha = 1.0$
12	I	714	12	102	A* with $\alpha = 0.3$
12	I	384	27	158	Breadth-First
12	II	374	12	106	A* with $\alpha = 1.0$
12	II	374	12	106	A* with $\alpha = 0.3$
12	II	372	30	179	Breadth-First
12	III	1072	9	178	A* with $\alpha = 1.0$
12	III	661	17	230	A* with $\alpha = 0.3$
12	III	403	59	314	Breadth-First

6.8 Conclusion

This chapter presented a search-based algorithm to select the best choice from a set of available alternatives in the strategic decision making problem with which the IC of a team is faced in multi-agent coordination. In addition, it estimates a time to indicate a minimum overall time of task execution that is required by the team to reach the goal through a sequence of strategic decisions. To expand the state space for finding the goal node, this algorithm generated new nodes by integrating two key algorithms for: 1) dynamic assignment of LoTeM tasks to agents to identify a right time for adaption of a SD and 2) calculation of a set of new alternatives for revising the SD of a node. To searhc the state space and select a node, this algorithm used three different search methods which include informed, breadth-first.

A spatial intelligent assistant system that is equipped with algorithm can support human decision and assists and collaborates with a human planner in spatial multi-agent planning/cooridnation. Results calculated by this algorithm are of considerable importance for the IC to 1) select the semi-optimal SD to constrain (partially specify) actions

of agents and 2) evaluate efficiency (qualify) of the defined strategy using the estimated total time and his intuition.

Optimization of a SD making problem is a NP problem especially in the domain of emergency management. This algorithm tries to solve this problem with a semi-optimal, feasible, complete solution in a shortest computation time.

An optimal plan, which comprises a sequence of strategic decisions, is calculated offline. But only the first SD of this plan that is one of the provided alternatives is of the considerable importance. Because of the uncertain and dynamic environment, this decision will be executed by agents online, and in real-time the IC should monitor the state of the world in order to identify a right time to adapt this decision to the new situation. It means that over time, this algorithms is required to re-making an optimal SD.

To calculate the the "h" variable, we take into account the LoTeM task environment. The results shown that this estimation is not exact in a complex problem that includes a complex strategy, a big team, and many LoTeM tasks. We are required an intelligent algorithm to calculate a correct quantity for this parameter. Although, the breadth-first search algorithm guaranties an optimal solution, it is impossible to use it for a complex problem because it needs a huge computation time.

The future work includes three directions: 1) develop a simulation for evaluation of teamwork among agents in task distribution according to human strategic decisions, 2) develop an intelligent software agent that can recommend a better strategy to the IC in order to adjust/refine the human strategy, and 3) develop an intelligent algorithm using machine algorithms to correctly calculate the variable h .

References

[1] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, and Pedro Szekely. "Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination." Journal of Software, 2014. Accepted.

[2] Bigley, Gregory A., and Karlene H. Roberts. "The incident command system: High-reliability organizing for complex and volatile task environments." Academy of Management Journal 44, no. 6 (2001): 1281-1299.

[3] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[4] Hunger, J. David, and Thomas L. Wheelen. Essentials of strategic management. New Jersey: Prentice Hall, 2003.

[5] MacEachren, Alan M., Guoray Cai, Rajeev Sharma, Ingmar Rauschert, Isaac Brewer, Levent Bolelli, B. Shaparenko, Sven Fuhrmann, and Hongmei Wang. "Enabling collaborative geoinformation access and decision making through a natural, multimodal interface." International Journal of Geographical Information Science 19, no. 3 (2005): 293-317.

[6] Maheswaran, Rajiv T., Pedro Szekely, and Romeo Sanchez. "Automated adaptation of strategic guidance in multiagent coordination." In Agents in Principle, Agents in Practice, pp. 247-262. Springer Berlin Heidelberg, 2011.

[7] Malone, Thomas W., and Kevin Crowston. "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) 26, no. 1 (1994): 87-119.

[8] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[9] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research, 2014. Accepted.

[10] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." arXiv preprint arXiv: 1401.0282 (2014).

[11] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC), vol. 4, no. 1, pp. 39-46, 2014.

[12] Nourjou, Reza, and Michinori Hatayama. "Simulation of an Organization of Spatial Intelligent Agents in the Visual C#.NET Framework." International Journal of Computer Theory and Engineering, 2014. Accepted.

[13] Russell, Stuart, and Peter Norvig. "Artificial Intelligence: A Modern Approach Author." Publisher: Prentice Hall Pa." (2009): 1152.

[14] Sander, Torsten, and Philipp Wehner. "On A* search with stopover areas." International Journal of Geographical Information Science 23, no. 6 (2009): 799-819.

[15] Zeng, W., and R. L. Church. "Finding shortest paths on real road networks: the case for A*." International Journal of Geographical Information Science 23, no. 4 (2009): 531-543.

Chapter 7

Simulation of an Organization of Spatial Intelligent Agents

7.1 Abstract

Problem: This chapter addresses an engineering challenge in simulation of multi-agent systems. Simulation of an organization of spatial intelligent agents is an important issue for implementation and evaluation of distributed algorithms. The research question is that how to simulate this organization using the Visual C#.NET.

Objective: This chapter intends to apply the C# programming language for development (or implementation) of a simulator. In a simulated environment, a social agent needs to communicate with other agents and response to received message.

Method: Our approach is to integrate the thread and delegate methods provided by the .NET framework. We easily enhance the architecture of the spatial intelligent agent in order to embed a number of them in a simulated society. To get better understand, a basic programming code in C# is presented to show how to implement a contract-net protocol (CNP) among three simple agents.

Results: Two results were achieved: 1) responses and actions which three agents did during a simulated CNP and 2) a simulated environment which contains distributed spatial intelligent agents that can interact with each other and with a human user.

Conclusion: The proposed methodology and the presented code provide a flexible and efficient framework for C# developers to develop, simulate, and evaluate a society of advanced software agents in the .NET platform.

7.2 Introduction

GICoordinator, as a GIS-based intelligent assistant system, assists an incident commander in coordination of a team of field units in the domain of disaster emergency response in especial in urban search & rescue operations [7]. This spatial intelligent system supports the IC with intelligent algorithms for action planning and task scheduling for centralized coordination of a team in a dynamic and spatial environment [8, 9]. Also, it applies a spatial database for geographic and location-based information management and uses GIS functions to support development of these spatial intelligent algorithms [6, 7]. The C#.Net programming language has been used to implement the core of the GICoordinator [7]. An IC is equipped with a computer that runs an instance of GICoordinator. A simple version of this agent has been developed for the field units [5].

A difficult challenge arises in this domain when there are several teams that each one has an IC. In order to maximize a joint objective, teams have to cooperate with each other and coordinate their actions that it is the main responsibility of incident commanders. As a result, distributed incident commanders form an organization or society. They needs to coordinate their decisions with each other in a decentralized approach. With regard to this requirement, the GICoordinator of an IC should provide two essential capabilities within this organization: 1) communicate and interact with other agents for data sharing and coordination, 2) use decentralized algorithms for coordinating the distributed decisions. Integration of a human with a GICoordinator is called a *human-agent team* in this chapter, and Fig. 7.1 presents an organization in which three human-agent teams are embedded.

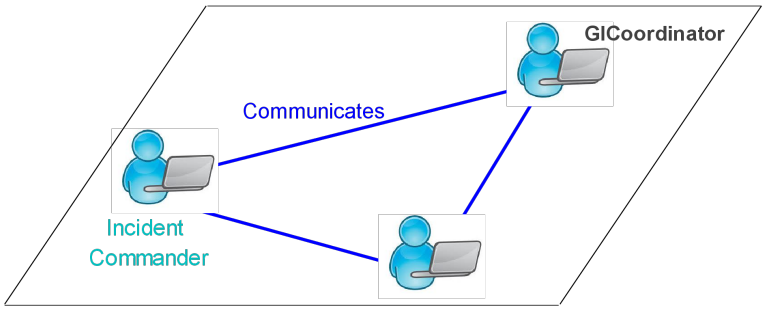


FIGURE 7.1: An organization of three human-agent teams

We require a proper framework to enable us to implement and evaluate two key capabilities identified for the GICoordinator before we develop real applications for the real world. We are faced with a problem in simulation of a society of advanced agents. Building this framework is a significant issue because we can easily run several GICoordinators in this framework, observe their behaviors, and test and refine the GICoordinator. We must consider that a single GICoordinator has been developed before.

Agent-based modelling and simulation (ABMS) is a relatively new approach to modelling systems composed of autonomous, interacting agents [4]. Simulation of an organization of GICoordinators is of considerable importance for developing and evaluating these two newly defined capabilities. A simulator provides us an efficient framework in which we can run a number of GICoordinators in a virtual world, study their interactions and behaviors in distributed problem solving, and test efficiency of distributed coordination algorithms. It enables us to easily refine their architectures and re-design and re-implement better algorithms.

In the intelligent systems technologies, there are cases in which the .NET framework is frequently used to develop an agent such as GICoordinator. The research question of this chapter is that how we can build a simulator using the C#.Net programming language. Our assumptions include as follows:

- communication between agents is done by sending and receiving messages.
- simulator and embedded agents are run in the same computer.
- each agent accesses to contents of a central spatial database.
- a human user can select any agent and start interacting with it.
- interaction between an agent and the human user is done via a user interface provided by this agent.
- description of distributed coordination algorithms is beyond the scope of this chapter.
- we can initiate and run any number of agents.
- we assume that there are three agents.
- agent have GIS functions
- the C#.Net platform should be used to create this simulator.
- a agent needs to send a specific message to a specific agent.
- a agent needs to appropriately respond to received messages.

In order to build the simulator, we need to apply an efficient tool according to the required assumptions. However, there is much literature in agent-based modeling software and toolkits, agent-oriented programming languages unfortunately they do not thoroughly address the problem stated by this chapter. As a result, we are faced with a difficult challenge.

This chapter intends to present a methodology for development of a proper simulator and simulation of an organization of distributed GICoordinators using the C#.Net programming language according to the defined requirements. In the simulated society, a GICoordinator can communicate with definite agents, response to received message, and interact with a human user.

7.3 Background

Agent-based modelling and simulation (ABMS) is a relatively new approach to modelling complex systems composed of interacting, autonomous 'agents'. Agents have behaviours, often described by simple rules, and interactions with other agents, which in turn influence their behaviours ABMS can be traced to investigations into complex systems. A typical agent-based model has three elements: 1) a set of agents, 2) a set of agent relationships and methods of interaction, and 3) the agents' environment. A model developer must identify, model, and program these elements to create an agent-based model [4].

In general, two types of simulation / modelling systems are available to develop agent-based models: toolkits or software [1]. Toolkits are simulation / modelling systems that provide a conceptual framework for organising and designing agent-based models. They provide appropriate libraries of software functionality that include pre-defined routines and functions specifically designed for ABM. However, the object-oriented paradigm allows the integration of additional functionality from libraries not provided by the toolkit, extending the capabilities of these toolkits. Some toolkits include: Swarm, MASON, Repast, OBEUS, AnyLogic.

In addition to toolkits, software such as StarLogo, NetLogo, OBEUS is available for developing agent-based models, which can simplify the implementation process. For example, simulation / modelling software often negates the need to develop an agent-based model via a low-level a programming language (e.g. Java, C++, etc). In particular, software for ABM is useful for the rapid development of basic or prototype models. However, modellers using software are restricted to the design framework advocated by the software. For instance, some ABM software will only have limited environments (e.g. raster only) in which to model, or agent neighbourhoods might be restricted in size. Furthermore, a modeller will be constrained to the functionality provided by the software (unlike ABM toolkits modellers will be unable to extend or integrate additional tools), especially if the toolkit is written in its own programming language (e.g. NetLogo). Most simulation packages claimed that they are object oriented or use Java as development language.

An agent-based model could be programmed completely from scratch using a low-level programming language such as s Python, Java, and C. These languages also can be used, but development from scratch can be prohibitively expensive given that this would require the development of many of the available services already provided by specialized agent modelling tools. Most large-scale agent-based models use specialized tools, toolkits, or development environments based on reasons having to do with usability, ease of learning, cross-platform compatibility, and the need for sophisticated capabilities to connect to databases, graphical user interfaces and GIS [4]. In particular, the use of toolkits can reduce the burden modellers face programming parts of a simulation that are not content-specific [1].

7.4 Methodology

Fig. 7.2 presents the structure of the simulation problem which this chapter addresses. It is designed based on the defined requirements.

Our focus is on the organization simulated in the .NET framework. To implement this element, we used and integrated the thread and the delegate methods that the C#.NET framework includes. These technologies enable us to easily refine the architecture of the GICoordinator in order to achieve the purpose defined by this chapter.

A delegate is a type that defines a method signature. When you instantiate a delegate, you can associate its instance with any method with a compatible signature. You can invoke (or call) the method through the delegate instance. Delegates are used to pass methods as arguments to other methods. Event handlers are nothing more than methods that are invoked through delegates. You create a custom method and a class such as a windows control can call your method when a certain event occurs. Threading enables the C# program to perform concurrent processing so that you can do more than one operation at a time [2].

C# supports parallel execution of code through multithreading. A thread is an independent execution path, able to run simultaneously with other threads. We can write applications that perform multiple tasks at the same time. Tasks with the potential of holding up other tasks can execute on separate threads, a process known as multithreading or free threading [2].

To get a better understand, a fundamental code was developed in C# that it shows how to implement a contract-net protocol (CNP) among three simple agents in a simulated organization. This code is shown in Listing 1. CNP is a task-sharing protocol in multi-agent systems, consisting of a collection of nodes or software agents that form the

'contract net'. When a node gets a composite task (or for any reason cannot solve the present task) it breaks the problem down into sub-tasks and announces the sub-task to the contract net acting as a manager. Bids are then received from potential contractors and the winning contractor(s) are awarded the job(s) [**?**].

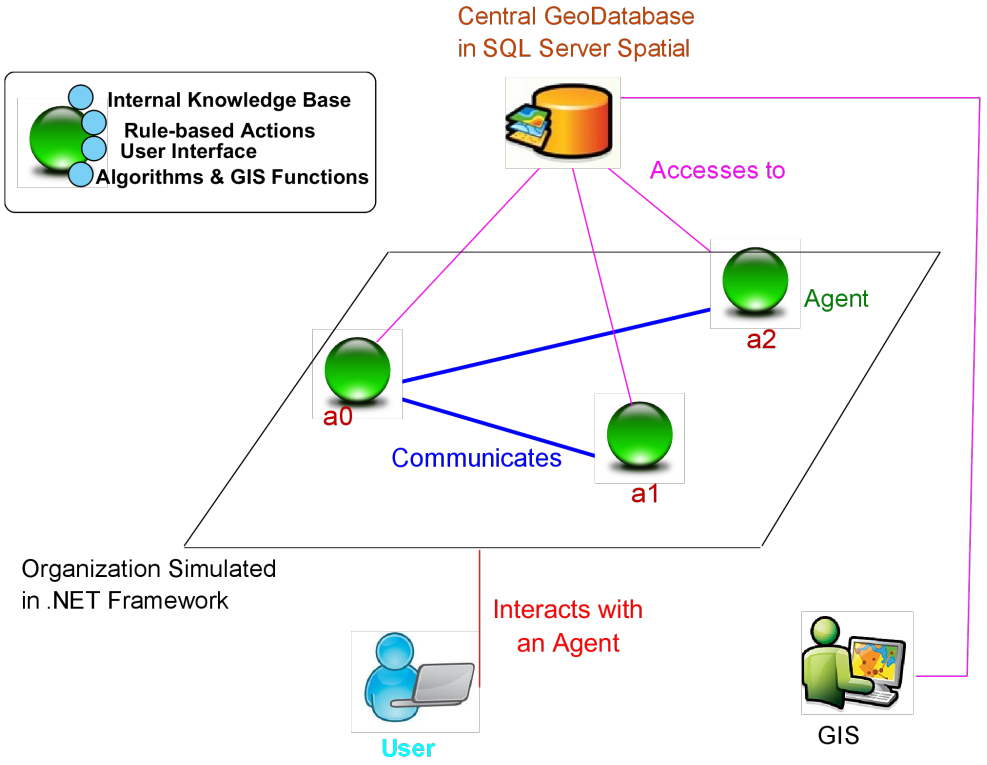


FIGURE 7.2: Structure of the simulation problem addressed by this chapter.

```
using System;
using System.Threading;
using System.Collections.Generic;

namespace SIMULATOR
{
    delegate void d_Send_f0t1(message msg);
    delegate void d_Send_f0t2(message msg);
    delegate void d_Send_f1t0(message msg);
    delegate void d_Send_f2t0(message msg);

    class Simulation
    {
        static void Main(string[] args)
        {
            Agent0 a0 = new Agent0("a0");
            Agent1 a1 = new Agent1("a1");
            Agent2 a2 = new Agent2("a2");

            a0.f_Send_t1 = new d_Send_f0t1(a1.f_Recieve_f0);
            a0.f_Send_t2 = new d_Send_f0t2(a2.f_Recieve_f0);
```

```
        a1.f_Send_t0 = new d_Send_f1t0(a0.f_Recieve_f1);

        a2.f_Send_t0 = new d_Send_f2t0(a0.f_Recieve_f2);

        Thread T_a0 = new Thread(a0.Run); T_a0.Start();
        Thread T_a1 = new Thread(a1.Run); T_a1.Start();
        Thread T_a2 = new Thread(a2.Run); T_a2.Start();

        Thread.Sleep(30000);
    }
}
class Agent0
{
    public string agentId;
    public int count;
    private List<message> Messages;
    public d_Send_f0t1 f_Send_t1;
    public d_Send_f0t2 f_Send_t2;

    public Agent0(string id) { agentId = id; }
    public void Run()
    {
        Thread T0 = new Thread(f_ReactiveRules);
        T0.Start();

        message msg = new message
       {
           from = agentId,
           to = "ALL",
           subject = "Announcement-a-contract",
       };
        count = 0;
        Messages = new List<message>();
        var T1 = new Thread(() => f_Send_t1(msg)); T1.Start();
        var T2 = new Thread(() => f_Send_t2(msg)); T2.Start();
    }
    public void f_Recieve_f1(message msg)
    {
        Console.WriteLine(agentId+": f= "+msg.from+" :Bid= "+msg.content);
        Messages.Add(msg);
        count = count + 1;
    }
    public void f_Recieve_f2(message msg)
    {
        Thread.Sleep(1000);
        Console.WriteLine(agentId+": f= "+msg.from+" :Bid= "+msg.content);
        Messages.Add(msg);
        count = count + 1;
    }
    private void f_ReactiveRules()
    {
        /* Monitor the enviroment and Do something for Example*/
        while (true)
        {
```

```
            if (count != 2) { Thread.Sleep(1000); continue; }

            List<message> Winners = new List<message>();

            if (Messages[0].content > Messages[1].content)
            { Winners.Add(Messages[0]); }
            else if (Messages[0].content < Messages[1].content)
            { Winners.Add(Messages[1]); }
            else if (Messages[0].content == Messages[1].content)
            { Winners.Add(Messages[0]); Winners.Add(Messages[1]); }

            message award = new message
            {
                from = agentId,
                subject = "Award",
            };
            foreach (message winner in Winners)
            {
                if (winner.from == "a1")
                {
                    var T1 = new Thread(() => f_Send_t1(award)); T1.Start();
                }
                if (winner.from == "a2")
                {
                    var T2 = new Thread(() => f_Send_t2(award)); T2.Start();
                }
            }
            break;
        }
    }
}
class Agent1
{
    public string agentId;
    public d_Send_f1t0 f_Send_t0;

    public Agent1(string id) { agentId = id; }
    public void Run()
    {
        /* Do something */
    }
    public void f_Recieve_f0(message msg)
    {
        Thread.Sleep(1000);
        Console.WriteLine(agentId+": f= "+msg.from+" :Sub= "+msg.subject);

        if (msg.subject == "Announcement-a-contract")
        {
            message msg2 = new message
            {
                from = agentId,
                to = "a0",
                subject = "Bidding",
                content = (new Random()).Next(1, 3)
            };
```

```
                var T1 = new Thread(() => f_Send_t0(msg2)); T1.Start();
            }
            if (msg.subject == "Award")
            {
                Console.WriteLine(agentId + ":: A Nice Day");
            }
        }
    }
    class Agent2
    {
        public string agentId;
        public d_Send_f2t0 f_Send_t0;

        public Agent2(string id) { agentId = id; }
        public void Run()
        {

        }
        public void f_Recieve_f0(message msg)
        {
            Thread.Sleep(1000);
            Console.WriteLine(agentId+": f= "+msg.from+" :Sub= "+msg.subject);

            if (msg.subject == "Announcement-a-contract")
            {
                message msg2 = new message
                {
                    from = agentId,
                    to = "a0",
                    subject = "Bidding",
                    content = (new Random()).Next(1, 3)
                };
                var T1 = new Thread(() => f_Send_t0(msg2)); T1.Start();
            }
            if (msg.subject == "Award")
            {
                Console.WriteLine(agentId + ":: A Nice Day");
            }
        }
    }
    class message
    {
        public string from { get; set; }
        public string to { get; set; }
        public string subject { get; set; }
        public int content { get; set; }
    }
}
```

LISTING 7.1: The C# Code for Implementation of a CNP in a Simulated Organization of Three Simple Social agents

To establish a communication network among three agents, four delegates were declared. As Fig. 7.3 presents, there are two methods to setup a communication network among

agents. An agent can communicate with another agent via an independent delegate, or there is a shared delegate that enables all other agents to send their message to this agent via this delegate. The following example shows a delegate declaration that is used by the agent a1 to send its messages to the agent a0:

```
delegate void d_Send_f1t0(message msg);
```

FIGURE 7.3: Two possibilities for configuration of a communication network.

Communication is used by agents to share or exchange data and information. To do it, the class "message" is declared to encode three kinds of data. We can design a number sophisticated classes for sending different types of data.

To present agents, three classes were declared. The architecture of an agent includes 1) a number of delegates for sending its messages to others, 2) a number functions for receiving messages from others, and 3) other properties and methods. Finally, three agents were created.

To setup a communication channel between two agents e.g. between the agent a0 and a1, the following code is the importance. It states that the agent a0 can send a message to the agent a1 if it calls the the function "f_Send_t1", and the agent a1 receives a message from the agent a0 through the function "f_Recieve_f0". In fact, the agent a0 executes a method of the agent a1.

```
  a0.f_Send_t1 = new d_Send_f0t1(a1.f_Recieve_f0);
```

To create three autonomous agents, the newly created agents are run via the three threads. The following code shows a thread which is created for the method "run" of the agent a0:

```
Thread T_a0 = new Thread(a0.Run);
T_a0.Start();
```

A proper delegate should be run by an agent to send a message to a certain agent. The agent a0 executes the following delegate in order to send a message to the agent a1.

```
var T1 = new Thread(() => f_Send_t1(msg));
T1.Start();
```

The function f_ReactiveRules of the agent a0 enables this agent to autonomously monitor and scan the environment. In this code, the responsibility of this function is to

continuously scan the total number of bids. If this agents gets all proposals from two agents, a simple sub-algorithm will be run to select winner(s).

7.5 Result

Two results were achieved in this chapter. If we run the code presented in Listing 1, we will get the result shown in Fig. 7.4. It presents responses and actions which three agents did during a simulated CNP.

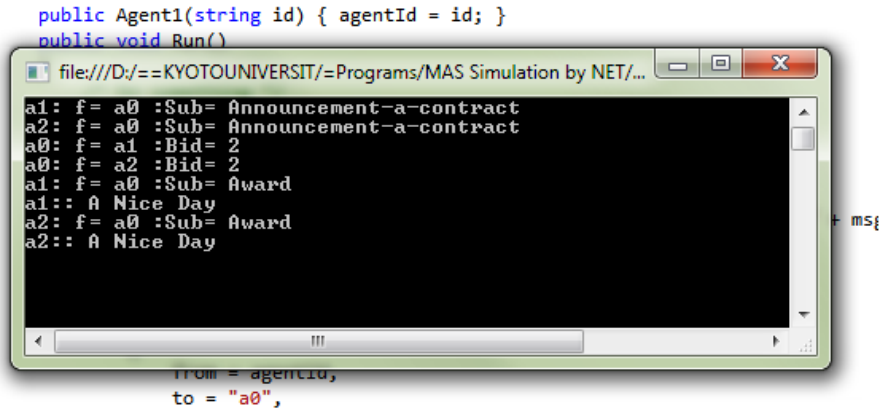


FIGURE 7.4: Result of running the Listing 1

Fig. 7.5 shows the second result. It is a simulated environment which contains distributed spatial intelligent agents that can interact with each other and with a human user. We enhanced the code shown in Listing 1 to achieve the purpose of this chapter according to the structure shown in Fig. 7.2. Each GICoordinator has its own user interface that enables the human user to select it and interact with it.

7.6 Conclusion

This chapter presented a C# code that can simulate a virtual society of distributed agents using the .NET framework. This basic code presented how to implement a communication network among social agents.

The proposed methodology provides us a flexible framework to develop, simulate, and evaluate multi-agent systems in the .NET platform. It enables us to work on decentralized algorithms for distributed GICoordinators by embedding a number of these agents in a simulated environment and studying/observing their actions.

For a mutual communication between two agents, we dedicated two delegates for this subject. It means that each agent has its won communication channels. It is possible

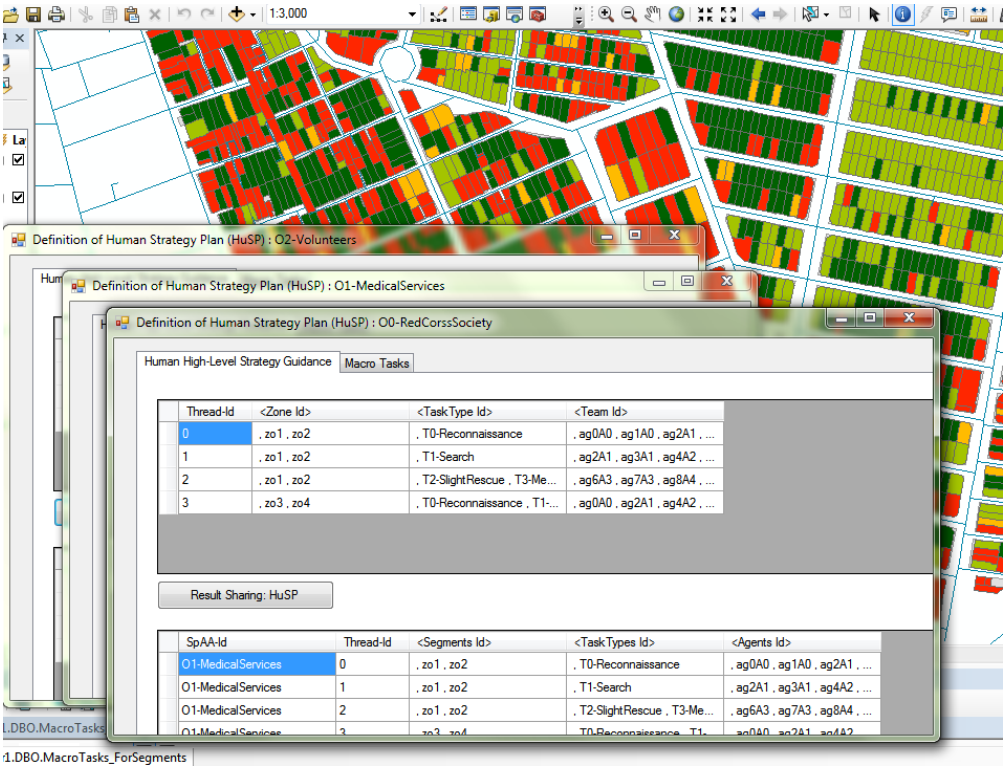


FIGURE 7.5: A simulated organization of three distributed GICoordinators

to specify a delegate for an agent and share it among others so that other agents can send their message to this agent via a shared delegate (or communication channel). The multithreading method allowed us to execute several agents and several actions in the same time.

Future work will be to design and develop distributed algorithms for coordinating distributed decisions among incident commanders in disaster emergency response operations. Certainly, the simulated framework will be used in this research.

References

[1] Castle, Christian JE, and Andrew T. Crooks. "Principles and concepts of agent-based modelling for developing geospatial simulations." (2006).

[2] Microsoft, Visual C#, http://msdn.microsoft.com/en-us/library/vstudio/kx37x362.aspx.

[3] Davis, Randall, and Reid G. Smith. "Negotiation as a metaphor for distributed problem solving." Artificial intelligence 20, no. 1 (1983): 63-109.

[4] Macal, Charles M., and Michael J. North. "Tutorial on agent-based modelling and simulation." Journal of Simulation 4, no. 3 (2010): 151-162.

[5] Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. "Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions." In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

[6] Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. "Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team." Journal of Disaster Research, 2013. (in press)

[7] Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. "Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations." In Workshop on Robots and Sensors integration in future rescue INformation system, ROSIN'13. In conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'13), 2013.

[8] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. "Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems." International Journal of Machine Learning and Computing (IJMLC). (in press)

[9] Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, and Pedro Szekely. "Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination." Journal of Software, 2014. (under review)

Chapter 8

Simulation of Intelligent Mobile GIS

8.1 Abstract

Earthquake emergency response requires different number of teams to cooperate with each other and coordinate their activities to achieve global objectives. Coordination is the key challenging problem that field teams face in urban search and rescue (USAR) emergency response because of the geographic and uncertain environment. Therefore, it is necessary to optimize coordination of teams by allocating tasks to teams in time and space efficiently and sufficiently. The present chapter aims to propose an efficient approach that allows humans to collaborate with coordinator assistant agents to assign tasks to teams. To achieve this purpose, we did the following stages: (1) the structure of the USAR task force and environment characteristics were analyzed, (2) we analyzed the spatial coordination problem and then modeled it, (3) features and properties of required approach were assessed, (4) we designed a contract net based model for distributed spatial task allocation and proposed the architecture of spatially distributed intelligent assistant agents (SpDI2A), and (5) finally, our approach was implemented and validated in a geospatial simulation which was developed using AnyLogic software and Java programming. The final result composes of 10 features satisfying the features of the required system.

8.2 Introduction

USAR is the emergency response function which deals with the collapse of man-made structures [17]. USAR operations in earthquake emergency response require different

number of teams to have cooperation with each other and coordinate their actions to do complex tasks in order to achieve global utility and objective. USAR has become an important issue in the recent decades because a number of earthquake disasters have occurred and had many deaths and injured especially in the developing countries. USAR operations are vital in earthquake emergency response especially during the first 72 hours. Their main objective is to maximize number of rescued victims in the shortest time. To reach this goal, one appropriate solution is to make decision on allocating tasks to teams in time and space efficiently and sufficiently. Because of complex environment characteristics of earthquake disaster, coordination is a key challenging problem that field teams face during field operations.

Five reasons require coordination among teams in multi-agent systems: (1) preventing anarchy or chaos, (2) efficiency, (3) meeting global constraints, (4) distributed information, expertise or resources, and (5) dependencies between the teams' activities [18]. In coordination theory, coordination is the act of managing interdependencies between activities performed to achieve a goal [16]. In the recent decade, multi-agent systems have been used for disaster management systems and emergency response simulations [9], [**?**] that we can mention DEFACTO [20], ALADDIN [12], RoboCup rescue simulation [14], and Coordinators [15]. The main methods [8] of coordination problem solving are classified into the task allocation, coalition formation, multi-agent planning, communication and data sharing, and negotiation.

Characteristics of environment of emergency response make coordination of teams so difficult and complex. Emergency response requires an appropriate approach in order to overcome this challenge and optimize the objective of USAR. After required properties of this approach were analyzed in this chapter, we found the Geospatial feature vital. The present chapter aims at proposing this efficient approach for solving the coordination problem of USAR in the geographic and uncertain environment. This approach provides a framework that allows a human to collaborate with his coordinator assistant agents to assign tasks to field teams. We called this approach SpDI2As or agent. The chapter proposes the roles of the agents and the way for designing them. The chapter also illustrates its implementation ability and validity by developing a geospatial simulation.

8.3 Issues in Urban Search and Rescue Operations

8.3.1 Structure of USAR Task Force in Earthquake Disaster

USAR Task Forces [7], [21] are the key responsible organization to coordinate and carry out tasks and Fig. 8.1 shows its organization. They include the multiple teams of

planning (situation assessment), hazmat, search, rescue, medical, and logistics. An incident commander (IC) commands their activities and makes tactical decisions which determine what team has to do what task, where, when, and with whom. We will be discussed the workflow of USAR in the Section III.A for 2 types of teams: search and rescue.

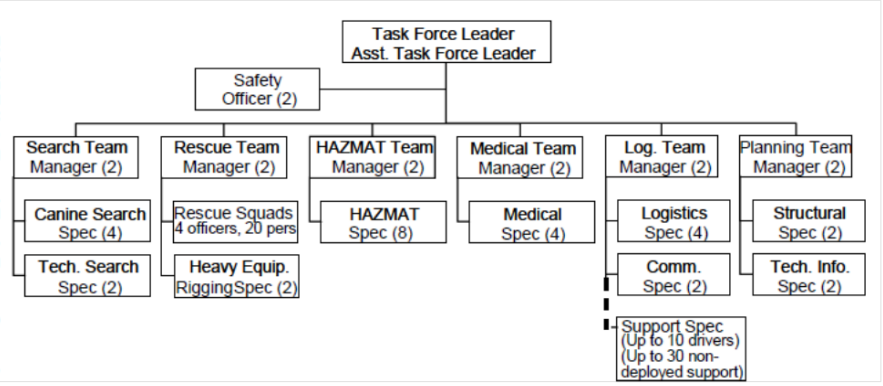


FIGURE 8.1: Organization of the USAR Task Force with 6 different field teams [7]

8.3.2 Environment Characteristics of Emergency Management

To understand the coordination problem of USAR teams clearly, we aimed to study and analyze environment characteristics of earthquake emergency response. We extended the results of other researchers [4], [13] to describe the following characteristics:

8.3.2.1 Key elements of environment

Tasks, teams, damaged buildings, and tactical decisions constitute key elements, which are shown with their related information in Fig. 8.2. In addition, treatment centers, operational areas, city blocks, and street network seem to be important.

8.3.2.2 Uncertainty and ambiguous information

This characteristic causes many challenging problems especially during the first 72 hours. For example, different information is reported to database; some tasks are revealed, done, or canceled unexpectedly; teams can not follow made decisions schedules; and database is not updated with new situation information.

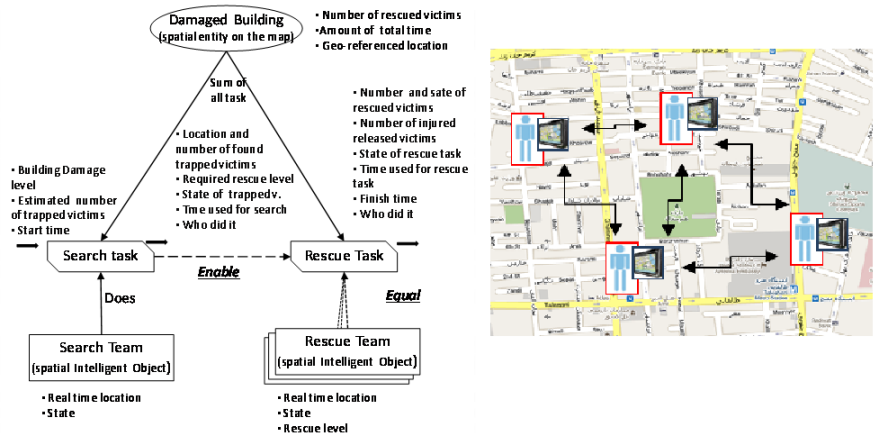


FIGURE 8.2: Geographical distribution of human-agent teams in area and collaboration human with SpDI2A via PDA; and spatial coordination problem of USAR modeled for one damaged building.

8.3.2.3 Geographic aspect

Earthquake disaster- affected area is an extensive geographical environment whose elements are spatial entities and are distributed in the environment.

8.3.2.4 Spatial decision making

Geographical distance or travel time among elements is an important parameter in action planning and decision making.

8.3.2.5 Dynamic in time and space

Teams have dynamic positions, and also information of elements varies over time.

8.3.2.6 Heterogeneous teams

Search teams and rescue teams, who are the 2 key teams, have different expertise and goals.

8.3.2.7 Multiple agents

There are more than one rescue teams or search teams at a USAR Task Force.

8.3.2.8 Real time decision making

Because of dynamic information, decision making should be done in real time way.

8.3.2.9 Interdependency Among activities

Tow types of interdependencies were studies in this chapter: "Enable" and "Equal". The "Enable" relationship specifies that when an action is carried out, it makes possibility of performing another action; and equality relationship means that certain actions are not liked to a specific team, and can be carried out by another team. As Fig. 8.2 shows, when the search task of damaged building is done by a responsible search team, the team manager reports new situation information such as location and number of found victims. The first interdependency says that this information reveals or discovers a new rescue task and enables rescue teams to have enough data for starting new rescue operations for the mentioned building. The second interdependency is that the new rescue task can be allocated to any suitable rescue team or teams not only to a certain team.

8.3.2.10 Distribution of expertise and information

Because of different capabilities of search teams and rescue teams, they have a cooperative relationship to complete complex tasks and reach a shared global utility.

8.3.2.11 Global objective

The final and global goal for all teams is to maximize number of rescued victims within a limit of time.

8.3.2.12 Geographical constrains

For each team, an operational region, which is a polygon or a zone, is defined. Each team is responsible for carrying out tasks which geographically are allocated in this area. For example, the incident commander asks their rescue teams to do rescue tasks of the central part of the city.

8.3.2.13 Ill- structure decision making problem

It is difficult to formulate the coordination problem with clear mathematical algorithms in order to optimize.

8.3.2.14 Partial observation

Teams have access a part of information of environment.

8.3.2.15 Failure of communication

Communication networks are not trustable and may fail during emergency response. It causes teams to lose their connection with the incident command post or with other teams.

8.3.2.16 Huge of data

Huge amount of data are gathered by many organizations with different standards.

8.3.2.17 Time constrains

Some tasks have deadline, some tasks should be done simultaneously, some teams should do a task simultaneously as a coalition, and some tasks should be done before others.

8.3.2.18 Limited resources

There are a few teams for doing a lot of tasks.

8.3.2.19 Spatial information sharing

Because of spatial aspect of environment, spatial information sharing among or teams is considered as the key property.

8.3.2.20 Centralized or decentralized coordinator

Sometimes decision making is done by a central incident commander for all teams, and sometimes it is done by several ones by distributed decision makers.

8.4 Approach

8.4.1 Analyzing and Modeling the Spatial Coordination Problem

In order to make decision and allocate tasks, first, we have to analyze and model the main problem. It will enable us to propose a solution for solving (optimize) it.

The described characteristics make USAR more complex; consequently, coordination among teams' actions gets a challenging problem, especially due to its spatial characteristic. We call this problem "Spatial Coordination Problem" (SCP).

In this section, we try to model the SCP for one damaged building, and collection of all buildings constitutes a SCP for whole USAR, as Fig. 8.2 shows. We try to model this problem by recognizing and analyzing teams, tasks, information flow, action interdependencies, attribute data, objectives, and workflow as following:

The global objective, which is maximize the number of rescued victims within a limit of time, is divided into 2 dependent sub- goals: search task and rescue task. To reach the goal, all sub-goals should be achieved, and if one fails, the whole utility will be zero. Data of location of the building and potential number of trapped victims are gathered by loss estimation teams. These data are input for search operations. Because exact location and exact number of trapped victims are un-known, so rescue teams can not do nothing and have to wait for results which will provided by search teams. Decision making is done for task allocation. The search task of building is assigned to a search team, and related information is sent to the search team. The search team moves toward the building and reaches it after amount of time. Then, the team starts to locate and find victims under debris. After amount of time, the team completes this task. Its result is output data which will be used for farther operations. Now, the team gets free and ready for another mission. Decision making determines that the rescue task of this building is assigned to a rescue team. The rescue team moves toward the building and reaches it after amount of time. Then, the team starts to release and rescue trapped victims. After amount of time, the team completes this task. Final result is data showing number of rescued victims, number of injured released victims. These data will be used for emergency medical transportation and medical teams.

8.4.2 Key Features of the Approach

To propose an efficient approach for solving the SCP, we clarified its key required features and as 10 following properties:

8.4.2.1 Solve and Model the main spatial coordination problem

The approach should deal with the problem well.

8.4.2.2 Personal assistant agent (PAA)

In real situation, each human (manager of a field team) is equipped with a PDA or a cell phone that runs a PAA, GPS, and wireless network. The agent is a personal assistant or software agent that aids its user (human) in decision making, coordination, communication with other PAAs, data management etc. Each agent is run on a PDA and collection of PDAs from the society of agents. Table 8.1 shows actions and behaviors of an agent. This feature leads to the concept of human-agent interaction. This concepts is shown at Fig. 8.2.

8.4.2.3 Mixed-initiative planning

Human can make strategy decisions for agent so that it takes into account them at decision making for task allocation. For example, in an emergency situation, a search team needs to assign a rescue task to a rescue team. He, therefore, should be able to define that task for his agent and request it to allocate this task to a rescue team. Moreover, human has actions and behaviors which are shown at Table 8.1. This feature leads to the concept of human- agent team and human-agent interaction. In fact, Human makes strategy decision in high level and agent makes tactical decisions based on those.

8.4.2.4 Human-agent interaction

Human-agent Interaction can be studied from four view points: 1) information acquisition; 2) information analysis; 3) decision and action selection; and 4) action implementation [5]. Because a human and his agent play a role in task allocation, it is necessary for them to interact together well. The interaction is done via the GIS-based interface. Based on human's actions, mentioned at Table 8.1, agent makes a correct action (decision) to response them. It can be implemented by Statechart diagram [22].

8.4.2.5 Communication among agents

Agents have to communicate together via exchanging messages, and they should understand each other. This property is necessary to solve the SCP.

TABLE 8.1: Action OF the Human agent and Software Agent and Their Interaction

Entity	**Actions / decisions**
Agent	1- Get different requests and messages from its human via user interface, and make decisions considering them 2- Retrieval, insert, query, update information of global geo-database 3- Retrieval, insert, query, update information of local geo-database 4- Display data (spatial and non-spatial) of geo-database (local and central) via maps and user-interface for human. 5- Communicate with other agents embedded in other PDAs in order to exchange messages, data, request etc. 6- Do analysis, calculation, GIS functions etc 7- Make right decision for messages received from the society 8- Make and send different kind of messages 9- Establish a contract net to allocate rescue tasks 10- Inform and display information of tasks which were assigned to the team
Human	1- Observe the real environment surrounding him and acquire information 2- Insert and modify information 3- Make and define strategy decision such as defining a rescue task for his agent in order to allocate it 4- Ask his agent to do geospatial analysis such as mapping buildings which are allocated in his operational area and their rescue tasks are not allocated 5- Reject a commitment to do a task 6- Move in area to reach buildings 7- Carry out tactical decisions (tasks) physically.

8.4.2.6 Methods of multi-agent coordination

This key feature is the core for SCP solving. Agents apply them for distributing tasks among teams.

8.4.2.7 Real time greedy algorithms

Methods of multi-agent planning or scheduling algorithms seem to be sufficient, but a very uncertain environment makes sequence decision process unreliable. Therefore, one

efficient solution can optimize local objectives and use greedy algorithms over time based on new updated data.

8.4.2.8 Decentralized decision making

Failure of network communication is one of the environment characteristics. Therefore, we can not rely completely on the centralized approach for commanding incident. Although distributed approaches do not undertake an optimal solution, they are efficient solution to overcome this characteristic. This lets teams have an autonomous role in decision making and allows them to allocate unsigned or revealed rescue tasks to rescue teams.

8.4.2.9 Geographic information management and data sharing

We consider two types of information management: central database and distributed (local) databases. Each agent has its own local database which is embedded in its PDA. All agents access the central database to improve data sharing. In real situation, central database can be a GIS server or Google maps [10].

8.4.2.10 GIS functions

Agents should have geospatial capabilities in mapping, analyzing, updating, sharing, and reasoning [1].

8.5 Methodology

8.5.1 Definition of the SpDI2A

An intelligent agent is an autonomous entity which observes and acts upon an environment and directs its activities towards achieving goals [19]. The important characteristics which an intelligent agent has in MAS include autonomy, local views, and decentralization [23]. An assistant agent, which is run on a PDA, can assist its human in doing some tasks such as scheduling, negotiating and communication with other agents, information analysis etc. If they are responsible for action coordination among teams, they are called coordinator agents. Agents which are housed and distributed in a georeferenced environment are called spatial agents, and they will possess 3D geographical location. As a result, SPDI2As contain all of these characteristics, and we call it agent

in this chapter. In summary, a society of agents can contain all of the properties which a required approach must have.

8.5.2 Method of Distributed Spatial Task Allocation

To implement the feature "The methods of multi-agent coordination", we applied the method of distributed task allocation [8] using contract net mechanism [21] for SCP solving by distributing rescue tasks among rescue teams. This mechanism is based on the decentralized market structure and consists of the 4 steps: (1) problem recognition, (2) tasks announcement, (3) bidding, and (4) awarding.

To distribute a rescue task among rescue teams, we designed a simple model and called it "distributed spatial task allocation". In our proposed approach, because search teams are autonomous entities and can distribute tasks among rescue teams. Imagine that a search task is completed by a rescue team and a new rescue task is discovered as its result. Based on the organizational structure, teams belonging to a same USAR Task force can communicate together. The following flowchart describes this model:

- A search task performed by a search team reveals a new rescue task. The human, user of PDA and coordinator of team, recognizes this task.
- On the GIS map provided by SpDI2A, human finds and locates the damaged building for which this rescue task is being defined; and then he inserts and modifies a rescue task with related information on the map; then he asks his agent to update central database and allocate it to the best rescue team.
- The agent receives this request from its human. To select a right action, it uses the component "Rule-based actions". To response this request, it uses the functions which are housed in the module "Methods and functions" and connects the central geo-database and updates its data. Then, it creates a new message "Rescue announcement" which contains the ID of concerned building, number of trapped victims, required rescue level etc. After that, the agent sends the message to all rescue teams' SpDI2As. Until a deadline (6 seconds), it is waiting for receiving proposals (bid) from enthusiastic rescue teams. The SpDI2A changes its internal sate to "Anticipant". This change leads to update the local geo-database of agent.
- When a rescue team' SPDI2A receives messages from the society of SPDI2As, it sends them to the module "Rule-based actions" for determining a right action. The right action for the message "Rescue announcement" is called "Bidding". To do it, the agent needs to use spatial functions of module "Methods and GIS

functions". The goal of this action is to calculate a bid for the announced rescue task. Fig. 8.3 shows the structure of module "Rule-based actions" for responding to this message.

- The SpDI2A of the search team, who has initiated this contract, receives messages from the society of SPDI2As and gathers them in a set. After the deadline finishes, it sends the set to the module "Rule-based actions" to assess messages which are related to the announced rescue task. The minimal bid is determined in order to allocate the announced task to its team. The SPDI2A creates a message "Award" and sends it to the determined rescue team.

- The SpDI2A of winner rescue team receives the message "Award". It updates internal information of team and tasks list. The information of allocated task is displayed for human via the interface, so that human knows what next task to do.

```
If  (msg type is "Rescue announcement") then
{
  If (
    (Rescue level of team >= f_RequiredRescueLevel(msg)) AND
    (Is state of team "Ready") AND
     (Is task geographically located at the operational region of
     the team)
    ) then
  {
    Bid= distance between real-time location of team and
      location of task;
    Make a new message "bid";
    Send the message to the initiator agent;
    Change state of team to "anticipant" and wait for 13 second
      to get response;
  } Else
    Refuse in contract ;
}
```

FIGURE 8.3: Algorithm of calculation of bid as a part of the component "Rule-based actions" for making right action for the message type "rescue announcement".

8.5.3 Architecture of the SpDI2A

Considering the features of required system and the model of distributed spatial task allocation, initial architecture of SpDI2A was designed with 6 key modules or components as following: "Communication with SpDI2As", "Human-agent interaction", "Central geo-database", "Local geo-database", "Rule-based actions", and "Methods and GIS functions". Relations between the components are shown at Fig. 8.4.

A central geo-database can contain geographic and non-spatial information of damaged buildings, city blocks, street networks, and tasks.

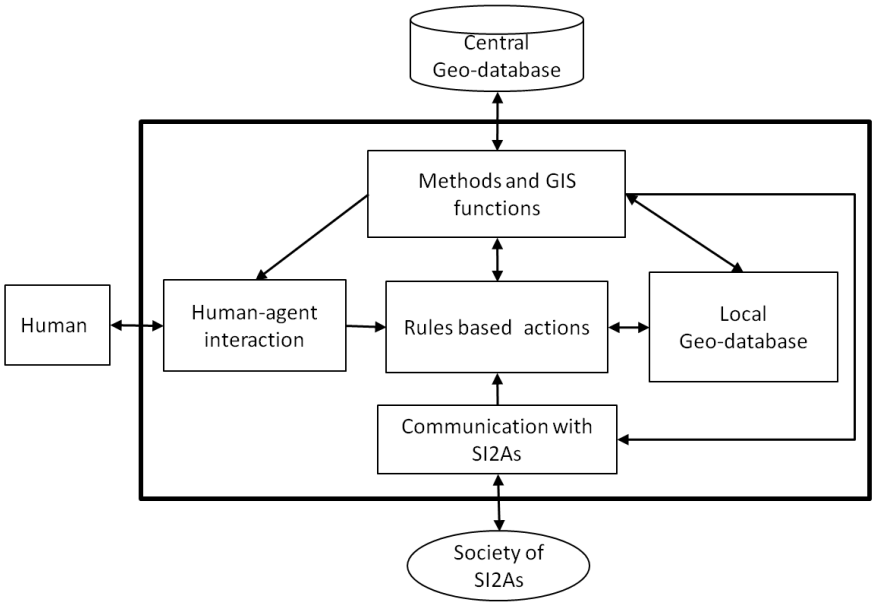


FIGURE 8.4: Architecture of SpDI2A.

Each agent comprises of a local geo-database. It contains local information such as real time position of team, team type, tasks list, messages, internal state (rest, busy, ready etc), operational region, ID of agents, city building, etc. The structure of local geo-database of rescue agents is different a little with that of search agent.

The component "Communication with SpDI2As" allows agent to communicate with agents of society via the message. This capability is necessary for agents to cooperate with others for task allocation. To do a certain action, maybe the agents create messages and send them to certain agents such as the message "Rescue announcement" and the message "Bidding".

The component "Rule-based actions" determines that what action agent has to select and do, as Table 8.1 shows these actions. When an agent receives a message from the society or from its human, the component "Rule-based actions" makes decision and determines actions to be done. This component has connections to other modules in order to carry out actions. Fig. 8.3 shows how agent makes decision for the message "Rescue announcement". To implement the distributed based spatial task allocation, we designed a state-based behavior both for search agent and for rescue agent, and then we implemented them using by the statecharts tools of AnyLogic [25]. Fig. 8.5 shows diagram developed for both agents.

The component "Methods and GIS functions" provides a laboratory of necessary functions and algorithms which agent needs for making actions, reasoning, rule-based decision making, calculation, analysis, and GIS functions [1]. These functions include data management (querying, retrieve, update, insert, store, connect to database) of the

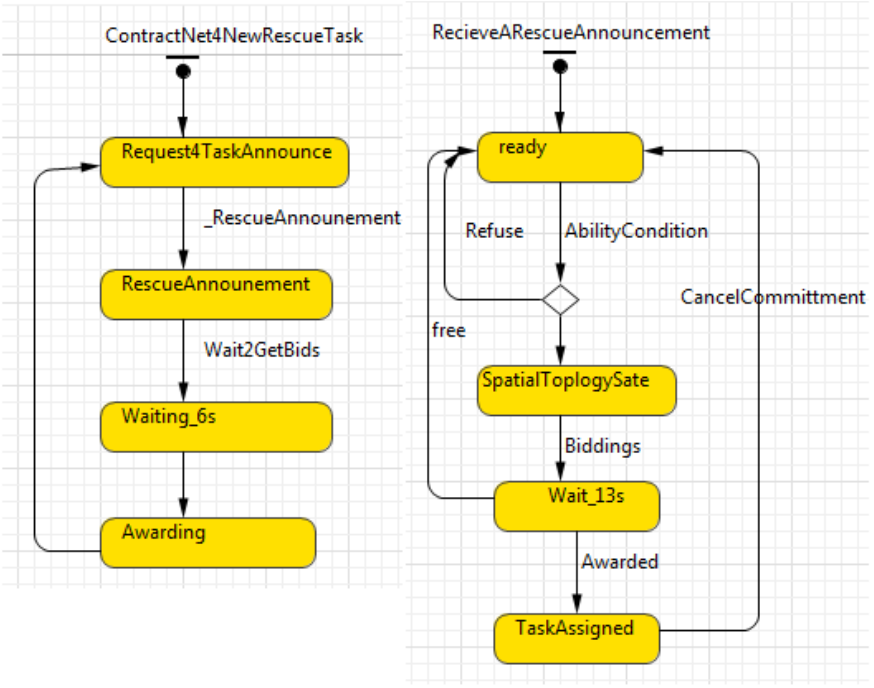


FIGURE 8.5: Sate-based behavior of agent (rescue and search) for implementation of the model of distributed spatial task allocation.

central and the local geo-database, bidding for a task, creating different messages, calculating distance between two point, acquire real-time position on the GPS of PDA, GIS analysis (spatial relationships, network analysis, proximity analysis), displaying information etc.

Human-agent interaction can be the interesting component of SpDI2A. Table 8.1 helps us to understand what decision and actions human and agent have. Interaction between human and agent is done via the developed user-interfaces, tools, and GIS maps which are run on screen of a PDA. Requests (actions) are sent by human via the user-interface to agent, and agent receives them and forwards them to the component "Rule-based actions". Information of central and local geo-database is displayed as GIS maps. For example, human of search team wants to identify a building on the map, modifies information of its rescue task, and asks agent to update geo-databases and allocate it to a rescue team. When agent of rescue team wins an announced rescue task, the information of allocated task is displayed for human and informs him about the new task. Also human has this role that can reject or cancel a commitment and asks agent to allocate it to another.

8.6 Geospatial Simulation of SPDI2A

Simulation is an appropriate approach to implement and validate a proposed approach. We preferred to implement a geospatial simulation because of these reasons: (1) this is efficient and controllable to validate, modify, and improve the architecture of SpDI2A, (2) within a virtual environment, communications among agents are done easily with the simulation software via messages, and we do not need to setup a wireless network, (3) The central geo-database can be implemented and housed in the same computer that agents are, and (4) a geospatial simulation aid us to implement a spatial-agent based modeling such as the model of distributed spatial task allocation.

There are a number of geo-simulation systems [3] for different applications. We chose the AnyLogic simulation system [24] because it matches most part of our needs and aids us to implement the proposed architecture of SPDI2A. AnyLogic is a java-based simulation platform for agent-based modeling, system dynamics modeling, simulation of discrete event systems etc. It provides tools for displaying GIS data, accessing to database, statecharts diagrams, exchanging messages, programming with Java, etc. To implement the proposed approach with a society of SPDI2As, we used AnyLogic, ArcGIS Desktop, and Java programming.

To implement and develop the central and local geo-database, we made GIS shapefiles ready using ArcGIS Desktop [6]. For case study region, we selected a part of the region 17 of city Tehran. We created three sample shapefiles (point and vector layer) for rescue agents, search agents, and damaged building with their attributes to display them on GIS map in the simulated environment. To extract spatial relationships between buildings with operational regions of agents, we applied the GIS functions of ArcGIS to make a analyzed table that allows agents to determine whether a damaged building is located within the mission area of team or not. To manage and access to data and information by agents, we implement a Microsoft Access database and wrote SQL functions for component “Methods and GIS functions”.

Fig. 8.6 shows the SpDI2As which is implemented and validate within this geospatial simulation; as well, it presents the environment that interaction between human and his agent happens. We make its features clear as following:

8.6.1 Simulated interface of PDA

This is the environment that interaction between human and agent happens. Every time, the simulator allows its user (human) to choose another team, who is displayed on

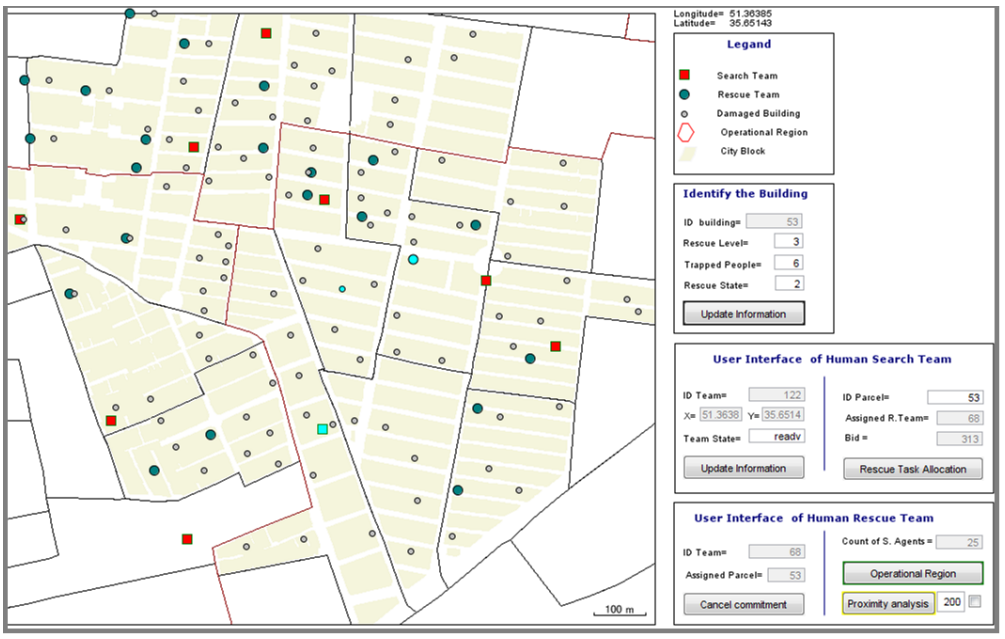


FIGURE 8.6: Geospatial simulation of SpDI2A and environment of human-agent interaction.

the map. As Fig. 8.6 shows, the user has chosen to be search team 122, and in another time, the role selected by the user was the rescue team 68.

8.6.2 Display information of the geo-database

Agent can visualize data of geo-databases (local and central) on the GIS map. Implementation of this feature is based the OpenMap [2], a geospatial visualization toolkit, which AnyLogic supports.

8.6.3 Local information management

Because every agent can access its local database, it can present internal information of team via the user-interface for its human, and human can modify them and ask his agent to update them. Furthermore, the agent displays location of team on the map with a symbol point. Fig. 8.6 shows this feature for search team 122. We should say that although the simulation shows position of all teams in the same time for the user, but the human will not see them in real situation based on PDA based implementation.

8.6.4 Information management

Because the central geo-database contains this information, each agent can access it to display, update, query, and retrieve its data as the functions of information management.

After human (human of search team 122) identifies a damaged building on the map, agent queries and extracts its information and its tasks from database and displays them into the interface; and then, human modifies them and asks agent to update the database. As Fig. 8.6 shows, this process happened for the damaged building 53 and the rescue task, which is recognized by human. This information is shared among all agents and will be used for the method of distributed spatial task allocation.

8.6.5 Rescue task allocation

Imagine that after the search team 122 completes the search task, located at the damaged building 53, a new rescue task is revealed. Human 122 recognizes it, and he tries to allocate it. It is enough to input the ID of damaged building into the user-interface and ask agent to allocate it. Agent receives this request from its human and establishes (initiates) the model of distributed spatial task allocation, discussed before. Finally, agent shows the final made decision for the announced task. As Fig. 8.6 shows, the task is allocated to the rescue team 68 who gave a bid of 313.

8.6.6 Management of task list

If the user of system switches from search team 122 to rescue team 68, his agent will inform him about a new task (tactical decision) which has allocated to his team. Fig. 8.6 shows a rescue task which is defined for the damaged building 53 is assigned to his team, and he and his teammates should move to this location to carry it out. If human wants to cancel his commitment, he can ask his agent to change the internal state and local information and allocate it to another rescue team too. Our assumption is that agent and human have the mutual trust, and agent does not control or monitor behaviors of its human. To find the location of the damaged building 53, agent highlights the symbol of this building for human.

8.6.7 Geospatial reasoning

We investigate role of GIS that enable agents to have skills for geospatial reasoning.

8.6.7.1 Proximity analysis:

Imagine that the rescue team 53 wants to see location of all damaged buildings which both are located in a distance of less 100 m and their rescue tasks are not assigned to

anyone. He requests agent to highlight these buildings on the map and calculates their total number, shown at Fig. 8.6. If the team moves, agent can make dynamic map regarding this request.

8.6.7.2 Spatial relationship analysis:

Again imagine that the rescue team 53 wants to see location of damaged buildings which are both located within operational area of team and their rescue task are not assigned to anyone. He requests his agent to highlight these buildings on the map and calculate and display the total number, shown at Fig. 8.6.

8.7 Discussion and Conclusion

SpDI2A is a geospatial approach that we tried to investigate the role of GIS in multi-agent coordination and propose a new generation of personal assistant agents that allow human to collaborate with coordinator assistant agent for coordination of emergency response. The 4 main features make SpDI2A distinctive from other works: 1) It has been designed for the organization of USAR Task Forces, 2) It has been proposed for USAR of earthquake disaster, 3) Geospatial features enable agents to have spatial intelligent and reasoning in dealing with spatial coordination problem in a geographic and uncertain environment, 4) Human-agent collaboration lets to plan teams' actions in time and space based on strategy decisions and tactical decisions, 5) SpDI2A can be used to design and develop "Intelligent Distributed GIS" that are run on the cell phones and assist field human teams for disaster management.

The architecture of SPDI2A is in its initial framework especially the method of distributed spatial task allocation which needs to get refined. Future works can focus on the: improving the task allocation method, accurate evolution of SpDI2A with other systems, applying strategy decisions for tactical decision making, and improvement of spatial data quality and spatial data sharing.

References

[1] Aronoff, Stan. "Geographic information systems: a management perspective." (1989): 58-58.

[2] BBN Technologies OpenMap,http://openmap.bbn.com

[3] Castle, Christian JE, and Andrew T. Crooks. "Principles and concepts of agent-based modelling for developing geospatial simulations." (2006).

[4] Chen, Rui, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Coordination in emergency response management." Communications of the ACM 51, no. 5 (2008): 66-73.

[5] Drury, Jill L., Jean Scholtz, and Holly A. Yanco. "Awareness in human-robot interactions." In Systems, Man and Cybernetics, 2003. IEEE International Conference on, vol. 1, pp. 912-918. IEEE, 2003.

[6] ESRI, ArcGIS version10, http://www.esri.com

[7] FEMA. "National Urban Search and Rescue (US&R) Response System." http://www.fema.gov/pdf/emergency/usr/usr_fog_sept_25_2003_color_final.pdf, 2013.

[8] Ferber, Jacques. Multi-agent systems: an introduction to distributed artificial intelligence. Vol. 1. Reading: Addison-Wesley, 1999.

[9] Fiedrich, Frank, and Paul Burghardt. "Agent-based systems for disaster management." Communications of the ACM 50, no. 3 (2007): 41-42.

[10] Gaber, Heba, Safaa Amin, and Abdel-Badeeh M. Salem. "Geospatial multi-agent system for urban search and rescue." In Proceedings of the 15th WSEAS international conference on Computers. 2011.

[11] United Nations Office for the Coordination of Humanitarian Affairs. "INSARAG Guidelines and Methodology." http://www.usar.nl/upload/docs/insarag_guidelines_july_2006.pdf, 2006.

[12] Jennings, Nick, Sarvapali D. Ramchurn, Mair Allen-Williams, Raj Dash, Partha Dutta, Alex Rogers, and Ioannis Vetsikas. "The ALADDIN project: Agent technology to the rescue." In Proceedings of the First Intl. Workshop on Agent Technology for Disaster Management. 2006.

[13] Khalil, Khaled M., M. Abdel-Aziz, Taymour T. Nazmy, and Abdel-Badeeh M. Salem. "Multi-agent crisis response systems-design requirements and analysis of current systems." arXiv preprint arXiv:0903.2543 (2009).

[14] Kitano, Hiroaki, Satoshi Tadokoro, Itsuki Noda, Hitoshi Matsubara, Tomoichi Takahashi, Atsuhi Shinjou, and Susumu Shimada. "Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research." In Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on, vol. 6, pp. 739-743. IEEE, 1999.

[15] Maheswaran, Rajiv T., Craig M. Rogers, Romeo Sanchez, and Pedro Szekely. "Human-agent collaborative optimization of real-time distributed dynamic multi-agent coordination." In Workshop 25: Optimisation in Multi-agent Systems, p. 49. 2010.

[16] Martial, F. von. Coordinating plans of autonomous agents. Springer-Verlag New York, Inc., 1992.

[17] Murphy, Robin Roberson. "Human-robot interaction in rescue robotics." Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 34, no. 2 (2004): 138-153.

[18] Nwana, Hyacinth S., Lyndon C. Lee, and Nicholas R. Jennings. "Coordination in software agent systems." British Telecom Technical Journal 14, no. 4 (1996): 79-88.

[19] Russell, Stuart Jonathan, Peter Norvig, John F. Canny, Jitendra M. Malik, and Douglas D. Edwards. Artificial intelligence: a modern approach. Vol. 74. Englewood Cliffs: Prentice hall, 1995.

[20] Schurr, Nathan, Janusz Marecki, John P. Lewis, Milind Tambe, and Paul Scerri. "The defacto system: Training tool for incident commanders." In AAAI, pp. 1555-1562. 2005.

[21] Smith, Reid G. "The contract net protocol: High-level communication and control in a distributed problem solver." Computers, IEEE Transactions on 100, no. 12 (1980): 1104-1113.

[22] Varró, Dániel. "A formal semantics of UML Statecharts by model transition systems." In Graph Transformation, pp. 378-392. Springer Berlin Heidelberg, 2002.

[23] Wooldridge, Michael. An introduction to multiagent systems. Wiley. com, 2008.

[24] XJ Technologies Company, Anylogic Professional 6.3.1 simulation software, http://www.xjtek.com/, 2008.

[25] XJ Technologies Company. "Designing state-based behavior: statecharts." http://www.xjtek.com/files/book/Discrete_events_and_Event_model_object.pdf

Chapter 9

Conclusion

9.1 Conclusion

This thesis proposed the GICoordinator for supporting decisions of IC in disaster response. This software agent collaborates with human in decision making for coordinating field unites. This supports IC in strategic planning, macro tasks assignment, scheduling, and automated adaption of these decisions in central multi-agent coordination. Seven contributions were achieved that each chapter was dedicated to each one.

1. Design of a GICoordinator and required functions were defined. We integrates geoinformatics with artifice intelligent techniques in order to provide sufficient tools for support of an IC for the coordination of the disaster crisis response. The key insight is (1) support human decisions with geo-spatial intelligent software system, (2) provide A.I. techniques for strategic planning, macro tasks assignment, scheduling, and automated adaption of these decisions in central multi-agent coordination, (3) involve human in the loop and enable collaboration between human and system for decision making.

   Future works will investigate role of this system in coordination of decisions of distributed incident commanders.

2. A SAP data model were designed. The SAP problem data aims to model the strategic planning problem in coordination of an emergency response team during emergency response management. It is required to develop intelligent software systems that collaborate with the humans to address the SAP problem by good strategy specification, optimally strategic action planning, and automated adaption. This data model is a focus on a specific problem, but it can be refined to address other ICs' requirements. It includes five new findings.

- First finding is to analyze the SAP problem and design the SAP data model for modeling the SAP problem data. The SAP problem data model is important and critical for develop any approach for strategic action planning in a team for coordination of disaster emergency response management.
- Second one is to model geospatial-temporal macro tasks by the "macroTask" and "temporalMacroTask" classes and integrate with other classes. They are used for five purposes: 1) present a set of tasks summarized and distributed in geographic objects, 2) extract and integrate tasks information from one geographic layer for another layer according to their spatial topological relationships, 3) organize and manage temporal changes in tasks and estimate their effects on dependent tasks, 4) create thematic maps and extract new information, and 5) use for strategic action scheduling.
- Third finding is to encode and formulate human high-level strategy guidance by the "thread" and the "strategy" classes and integrate them with the data model. These classes enable human agents engage in the planning process and collaborate with an automated information system. These two classes integrate human's intuition and initiative in the data model and enable the automated system to apply them for making an optimal strategic action plan.
- Next finding is the "threadAssignment" classes to model a high-level strategic action plan. This class constrains agents' behaviors by allocating them to threads.
- The "temporalMacroTaskAssignment" and the "legalAssignment" classes are the fifth finding. These classes present a strategic action schedule and are integrated with other classes to form a complete data model of the SAP problem.

Future work includes two directions. First work may be to improve the SAP problem with regard to new requirements or demands e.g. resource allocation problem, coordination of distributed teams, or integrating the SAP data model with field units decision support systems, The last idea is to integrate the spatial database of the GICoordinator with information systems and develop proper algorithms for data data/information extraction and integration in order to get required information for this system from distributed data warehouses.

3. Two intelligent algorithms that aim to 1) automated calculation of feasible alternative scenarios for selecting a strategic decision in certain time and 2) autonomously detecting a right time in which this strategic decision should be refined by releasing a set of identified agent from their threads. In fact, this algorithm autonomously

controls which agents should be released from their thread within a strategic decision. The role of human is to select a alternative as a strategic decision. Some findings include as follows

- A strategic decision constrains agents to threads but it does not assign tasks to agents. A strategic decision controls the domain of activities of agents.
- A coalition, which is a set of agents, is sufficient for a thread if it provides all capabilities required by all tasks (enabled tasks and not yet enabled tasks) located in this thread. A question that arises here is that which coalition among available coalitions can accomplish this thread faster than others? This question will be addressed a future work.
- A huge number of feasible alternatives are found for a case which contains a team with big size and a complex strategy. Selection of two coalitions that provide minimum or maximum capabilities for a thread reduce the run time and number of generated alternatives.
- Importance of this algorithm is to assist human and collaborate with him in the flow chart of centralized multi-agent coordination. This issue is so important and essential especially in a situation that the human strategy defines agents for several threads.
- This algorithm can be used in the multi-agent planning problem. A action plan made by the IC is a sequence of strategic decisions that states how a team of agents can achieve the goal.

The future will be to add other parameters and requirements to these algorithms in order to model and address other aspects of the problem.

4. Automated algorithm that aims to assign location-based temporal macro tasks to field unites that the result includes (I) a macro task/action schedule, (II) an overall execution time of tasks execution and, (III) a right time for revising a strategic decision. Some findings include as follow:

   - A schedule provides an I.C. a proper solution for the coordination problem in a team. Multi-agent scheduling partially specifies actions of rational agents and constrains them temporally and dynamically. the I.C. delegates agents to autonomously make their own tactical decisions(planning and scheduling) or adapt their activities under these macro decisions.
   - This system executes this algorithm till the current strategic decision is valid. As a results, this algorithm calculates a time in which this strategic decision has to be adapted and revised. This result as useful information has two

advantages: 1) It enables the I.C. to refine this strategic decision in a right time and 2) it is used in a search algorithm that estimates a makespan, calculate a complete solution, and find an optimal solution for multi-agent planning and scheduling.

- Macro tasks enables the system to model and present distribution of tasks in different geographic scales. Spatial topology between spatial objects enables the system to extract new views of tasks, and these tasks can be used for task assignment. Up to quality of data, an I.C. can select different geographic layers such as zones or buildings and allocate tasks to them. So the I.C. can specify and contain actions in different spatial accuracies.
- Information fusion algorithms extract new useful information from scheduling results. They provide an efficient situational awareness for an I.C. to percept states of global environment. They support human decisions by geospatial reasoning, geo-visualization, complex queries, etc. For example, they can answer these questions: What activities will done in a specific zone during a specific period? When a specific task will be done and by who? What is the task list of a specific agent?

Future works to improve the current contribution may include two ways:

- In some cases, the tasks environment contains tasks that may require synchronous capabilities. It means that more that one agent have to coordinate their actions in order to provide the required capabilities for doing a specific task simultaneously. It is important for an ideal algorithm to form a proper coalition comprising of suitable agents and assign this task to this group.
- Decentralized coordination of distributed schedules is a significant issue in where multiple teams are involved in performing tasks. It is possible that there are interdependencies between actions of teams. In order to maximize the joint objective, it is necessary to apply algorithms for coordinating distributed schedules which are made by each incident commanders.

5. A* search algorithm were presented. It aims to select the best choice from a set of available alternatives in the strategic decision making problem with which the IC of a team is faced in multi-agent coordination. In addition, it estimates a time to indicate a minimum overall time of task execution that is required by the team to reach the goal through a sequence of strategic decisions. To expand the state space for finding the goal node, this algorithm generated new nodes by integrating two key algorithms for: 1) dynamic assignment of LoTeM tasks to agents to identify a right time for adaption of a SD and 2) calculation of a set of new alternatives for revising the SD of a node. To searhc the state space and select a node, this

algorithm used three different search methods which include informed, breadth-first. Some findings include as follows

- A spatial intelligent assistant system that is equipped with algorithm can support human decision and assists and collaborates with a human planner in spatial multi-agent planning/cooridnation. Results calculated by this algorithm are of considerable importance for the IC to 1) select the semi-optimal SD to constrain (partially specify) actions of agents and 2) evaluate efficiency (qualify) of the defined strategy using the estimated total time and his intuition.
- Optimization of a SD making problem is a NP problem especially in the domain of emergency management. This algorithm tries to solve this problem with a semi-optimal, feasible, complete solution in a shortest computation time.
- An optimal plan, which comprises a sequence of strategic decisions, is calculated offline. But only the first SD of this plan that is one of the provided alternatives is of the considerable importance. Because of the uncertain and dynamic environment, this decision will be executed by agents online, and in real-time the IC should monitor the state of the world in order to identify a right time to adapt this decision to the new situation. It means that over time, this algorithms is required to re-making an optimal SD.
- To calculate the the "h" variable, we take into account the LoTeM task environment. The results shown that this estimation is not exact in a complex problem that includes a complex strategy, a big team, and many LoTeM tasks. We are required an intelligent algorithm to calculate a correct quantity for this parameter. Although, the breadth-first search algorithm guaranties an optimal solution, it is impossible to use it for a complex problem because it needs a huge computation time.

The future work includes three directions: 1) develop a simulation for evaluation of teamwork among agents in task distribution according to human strategic decisions, 2) develop an intelligent software agent that can recommend a better strategy to the IC in order to adjust/refine the human strategy, and 3) develop an intelligent algorithm using machine algorithms to correctly calculate the variable h .

6. A simulator that provides a flexible and efficient framework for C# developers who want to develop, simulate, and evaluate community of a number of instances of the GICoordinator. Some findings include as follows:

- The presented C# code can simulate a virtual society of distributed agents using the .NET framework. This basic code presented how to implement a communication network among social agents.
- The proposed methodology provides us a flexible framework to develop, simulate, and evaluate multi-agent systems in the .NET platform. It enables us to work on decentralized algorithms for distributed GICoordinators by embedding a number of these agents in a simulated environment and studying/observing their actions.
- For a mutual communication between two agents, we dedicated two delegates for this subject. It means that each agent has its won communication channels. It is possible to specify a delegate for an agent and share it among others so that other agents can send their message to this agent via a shared delegate (or communication channel). The multithreading method allowed us to execute several agents and several actions in the same time.

Future work will be to design and develop distributed algorithms for coordinating distributed decisions among incident commanders in disaster emergency response operations. Certainly, the simulated framework will be used in this research.

7. Simulate spatially distributed intelligent assistants that each one runs on a tablet computer and assists the field units in distributed task distribution/allocation among the team members. SpDI2A is a geospatial approach that we tried to investigate the role of GIS in multi-agent coordination and propose a new generation of personal assistant agents that allow human to collaborate with coordinator assistant agent for coordination of emergency response. The 4 main features make SpDI2A distinctive from other works: 1) It has been designed for the organization of USAR Task Forces, 2) It has been proposed for USAR of earthquake disaster, 3) Geospatial features enable agents to have spatial intelligent and reasoning in dealing with spatial coordination problem in a geographic and uncertain environment, 4) Human-agent collaboration lets to plan teams' actions in time and space based on strategy decisions and tactical decisions, 5) SpDI2A can be used to design and develop "Intelligent Distributed GIS" that are run on the cell phones and assist field human teams for disaster management.

   Future works will improve this system and implement it on the Mobile phone for real world. The architecture of SPDI2A is in its initial framework especially the method of distributed spatial task allocation which needs to get refined. Future works can focus on the: improving the task allocation method, accurate evolution of SpDI2A with other systems, applying strategy decisions for tactical decision making, and improvement of spatial data quality and spatial data sharing.

Publications

Journal Publications

5. Nourjou, Reza, Pedro Szekely, Michinori Hatayama, and Stephen F. Smith, *"A* Search Algorithm for Optimal Strategic Decision Making in Spatial Multi-agent Planning."* International Journal of Geographical Information Science, 2014. Under review.

4. Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, and Pedro Szekely. *"Intelligent Algorithm for Assignment of Agents to Human Strategy in Centralized Multi-agent Coordination."* Journal of Software, 2014. Accepted.

3. Nourjou, Reza, Pedro Szekely, Michinori Hatayama, Mohsen Ghafory-Ashtiany, and Stephen F. Smith. *"Data Model of the Strategic Action Planning and Scheduling Problem in a Disaster Response Team."* Journal of Disaster Research 9, no. 3 (2014):.

2. Nourjou, Reza, and Michinori Hatayama. *"Simulation of an Organization of Spatial Intelligent Agents in the Visual C#.NET Framework."* International Journal of Computer Theory and Engineering, IJCTE 2014 Vol.6(5), 2014.

1. Nourjou, Reza, Stephen F. Smith, Michinori Hatayama, Norio Okada, and Pedro Szekely. *"Dynamic Assignment of Geospatial-Temporal Macro Tasks to Agents under Human Strategic Decisions for Centralized Scheduling in Multi-agent Systems."* International Journal of Machine Learning and Computing (IJMLC) 4, no. 1 (2014): 39-46.

Refereed Conference Publications

3. Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. *"Design Requirements of Spatial Intelligent Coordinators for Incident Commanders."* Proceedings of the 11th International Conference on Information Systems for Crisis Response and Management, 2014. Accepted.

2. Nourjou, Reza, Michinori Hatayama, Stephen F. Smith, Atabak Sadeghi, and Pedro Szekely. *"Design of a GIS-based Assistant Software Agent for the Incident Commander to Coordinate Emergency Response Operations."* In Workshop on Robots and Sensors integration in future rescue INformation system (ROSIN' 13). In Conjunction of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS' 13), 2013.

1. Nourjou, Reza, Michinori Hatayama, and Hirokazu Tatano. *"Introduction to spatially distributed intelligent assistant agents for coordination of human-agent teams' actions."* In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 251-258. IEEE, 2011.

Papers in Preparation

4. Nourjou, Reza , et al. "Analysis of the Decentralized Coordination Problem among Distributed Incident Commanders in Distributed Strategic Planning and Scheduling."

3. Nourjou, Reza, et al. "Simulation of a Spatial Agent-based Model of Teamwork in a Human Team."

2. Nourjou, Reza, et al. "Automated Allocation of Location-based Resources under Human High-Level Strategy in Emergency Transportation Services."

1. Nourjou, Reza, et al. "Development of Distributed Intelligent Mobile GIS for Tasks Allocation in a Search & Rescue Team."